Status Report on Developments of C++ Mathematical Libraries

LCG Application Area Meeting, November 17, 2004

Lorenzo Moneta
Andras Zsenei
CERN/ PH-SFT

Outline

- Introduction
  - *Aim and requirements*
- Contents of C++ mathematical libraries
- Overall design and library organization
- Detailed design for core components
- Current status
- Tests and Validation studies
- Libraries for fitting and minimization (MINUIT)
- Summary and Conclusions

The MathLib SEAL Work package

- Aim is to provide coherent set of Mathematical Libraries to end-users and developers of LHC experiments
- Requirement to use the same core library in all environments
  - simulation, reconstruction and analysis
  - from C++ and interactively (Python, CINT) via C++ dictionary
- Avoid duplication and maintenance burden of similar libraries
- Collaboration with experiments and LCG projects (ROOT)

Mathematical Library Users

- Software developers of experiments writing simulation, reconstruction or event analysis applications
  - Use within experiment framework
- Analysis tools (developer and users)
  - ROOT, HippoDraw, JAS, etc…
- Physicists performing data analysis with stand-alone programs in C++
- Python interactive users
  - Use together with other tools provided in Python

General Requirements for Math Lib's

- Modularity : set of components with as little coupling as possible
- Allow dependency on C++ Standard Library
  - use std::vector and std::complex
- Refrain from duplicating functionality already present in STL
  - □vector operations, searching and sorting algorithms, etc..
- Avoid non - mathematical functionality
- Thread-safe (no use of static variables)
- Portability (at least on all LCG platforms)
- Allow dependency on external products if
  - they provide directly needed functionality
  - meet support and quality standard

Math Libraries Contents

- Required functionality is (non-exhaustive list):
  - Evaluation of Special and Statistical functions (Pdf)
  - Numerical Algorithms (Integration, Differentiation, Minimization, etc..)
  - Linear Algebra
  - Random Number generators and distributions
- Produced an inventory of functions and algorithms
  - group them by related functionality
  - with links to GSL, CERNLIB and ROOT documentation
  - available on the Web at:
    - » http://www.cern.ch/mathlib/mathTable.html

Inventory of Mathematical Functions and Algorithms

Functions and Polynomials	Numerical Methods	Random Numbers and Distributions	Others
• Special Functions • Polynomials • Function Approximations	• Integration • Differentiation • Minimization • Root-Finding • Interpolation	• Random Number Generator • Random Number Distribution	• Linear Algebra • Differential Equations • FFT

Special Functions

Routines for evaluating Special functions

Bessel Functions of various types

Regular cylindrical functions	*Bessel J functions of various orders*	GSL, Cernlib, R
Irregular cylindrical functions	*Bessel Y functions of various orders*	GSL, Cernlib, R
Regular modified cylindrical	*Bessel I functions of various orders*	GSL, Cernlib, R
Irregular modified cylindrical	*Bessel K functions of various orders*	GSL, Cernlib, R
Regular spherical functions	*Bessel j functions of various orders*	GSL, Cernlib
Irregular spherical functions	*Bessel y functions of various orders*	GSL, Cernlib

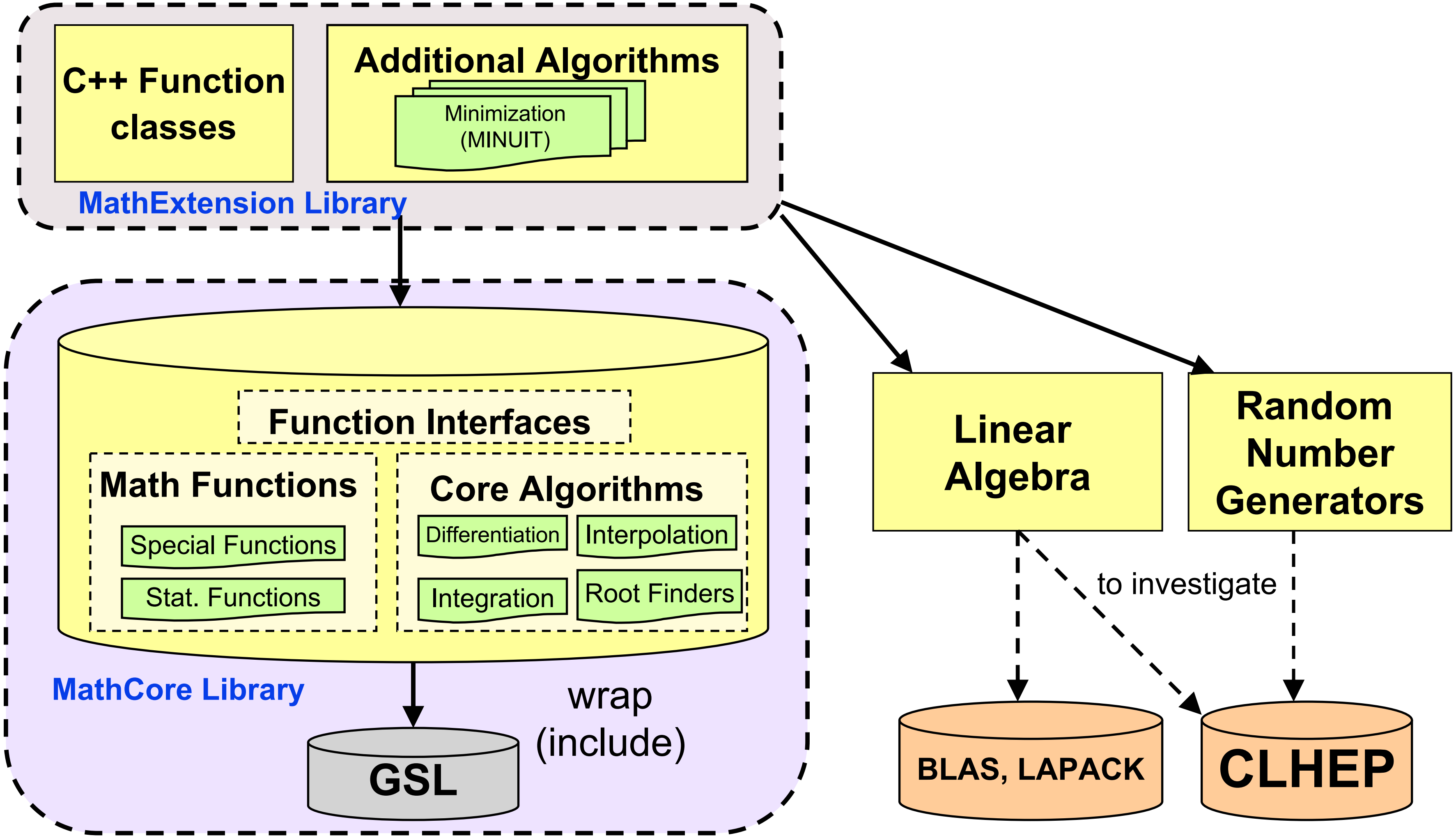
---

Clausen function	*Clausen integral function*	GSL, Cernlib
Coulomb Wave Function	*Wave functions for bound states and scattering solutions*	GSL, Cernlib
Dawson's integral function	*Dawson integral*	GSL, Cernlib
Dilogarithm function	*Dilogarithms for real and complex arguments*	GSL, Cernlib
Complete Elliptic integrals	*Legendre form of the various types of complete Elliptic integrals*	GSL, Cernlib
Uncomplete Elliptic integrals	*Carlson and Legendre form of uncomplete Elliptic integrals*	GSL, Cernlib (2)
Error functions	*Error function (ERFC) and complementary*	GSL, Cernlib, R
Exponential integrals	*Various type of exponential integrals*	GSL, Cernlib

C++ MathLib Components

- Mathematical functions
  - Special functions and statistical functions
  - Library of free (stateless) functions
- Algorithms (Core)
  - Numerical integration and differentiation, root finders, interpolation, function approximation
- Algorithms (2)
  - Function minimization and fitting, nonlinear root finding, differential equations, FFT, Monte Carlo integration
- Function classes
  - Generic function interface
  - Parametric functions, probability density functions (pdf)
  - Support for function operations (addition, composition, convolution)
- Linear Algebra
  - Vector and Matrices classes and their operations
- Random numbers
  - Generators engine and random distributions

Mathematical Libraries Organization



MathCore Library

- A Core Library with most used functionality:
  - Special functions
  - Statistical functions: distributions (pdf), probability (cdf) and inverse
  - Core algorithms: numerical integration and differentiation, root finders, interpolation, etc..
  - Basic function interfaces for all algorithms
- Exact content will be determined weighting size with respect to desired functionality
  - Constraint from ROOT to have size of order 1MB size
- The GNU Scientific Library (GSL) will be used NOW for implementing the majority of functionality
  - We will investigate using also other libraries (e.g. CEPHES for functions)
- Requirements from ROOT to have a standalone library (no external dependency)
  - Provide possibility to build the library including the GSL code
  - Use script to select and build automatically required code from GSL
- Current Status:
  - prototype available for testing ROOT integration

Math Extension Library

- One or maybe more libraries (to be seen)
- Sophisticated numerical algorithms like :
  - Minimization (MINUIT), global optimization algorithms (genetic alg.)
  - Multi Root finders, Monte Carlo integration, etc..
  - Dependency on Linear Algebra and/or Random Numbers
- Statistics utility:
  - Confidence level, quality of fits, comparison of distributions, etc…
- C++ classes for math functions and their operations
  - Arithmetic (+,-,*,/), composition and convolution
  - Convenient package to be used together with algorithms:
    - » e.g. Composing functions for fitting
- Provide means for user extensions
  - Define interfaces for algorithms
  - Possibility to introduce new algorithms and use at same level of others
    - » Use of abstract factories or plug-in manager

Linear Algebra

- Library with matrix and vector classes
  - use C++ operator overloading to implement vector/matrix operations
- Various implementations currently used in HEP (CLHEP, ROOT,....)
- Goal is first to evaluate and review existing packages
  - Performance studies in HEP application environments
- Developed a prototype based on expression templates:
  - Wrapper based on BLAS/LAPACK and GSL and used it for Linear Algebra studies comparing with CLHEP, Boost, ROOT
  - Have a version based on a customized implementation in MINUIT
- Next steps:
  - improve existing wrapper optimizing matrix allocation

Mathematical Libraries (cont..)

- Random Numbers Library
  - Library with generators and distributions
  - CLHEP is library mostly used currently in HEP
  - Investigate whether re-implement library following the design proposed for C++ standard
- Dictionary libraries for interactivity and persistency
  - Libraries generated using the LCG-ROOT dictionary
  - Produce for example Python bindings with PyLCGDict for function classes and Algorithms :
    - » easy use of the library in the interactive environment
  - Reflection information will allow persistency of complex objects (matrices, functions, random number seeds, etc…)

Mathematical Functions Design

- Evaluation of functions at a point
  - No need for objects, have a simple procedural API
- Set of free functions in a namespace
  - Approach adopted by C++ standard committee
    - » use same proposed name scheme
  - Advantages are (w.r.t to static function in a class) :
    - » Users can extend and add new functions in same namespace
    - » Users can overload them for new type of data
- Library hides detailed function implementation
  - Implementing majority of functions as wrapper to GSL
    - » introduce negligible overhead for these functions
  - Can replace implementation in the future without changes for user code

Mathematical Functions Contents

- Special Functions (~ 20 functions)

  - Bessel (various type)
  - Beta
  - Gamma
  - Zeta
  - Error functions
  - Elliptic integrals
  - Hypergeometric
  - Exponential integral
  - Legendre polynomials

- Statistical Functions: probability distributions (pdf), cumulative distributions (Q and P integrals of pdf) and their inverse:

  - Chi2
  - Landau
  - Gamma
  - Beta
  - LogNormal
  - Breit-Wigner
  - Student t distribution
  - Fischer distribution
  - Poisson
  - Binomial

Example of Free Functions

- Header file : MathCore/SpecFunc.h

```
namespace mathlib {

// [5.2.1.8] regular modified cylindrical Bessel functions
double cyl_bessel_i(double nu, double x);
// [5.2.1.9] cylindrical Bessel functions (of the first kind)
double cyl_bessel_j(double nu, double x);
// [5.2.1.10] irregular modified cylindrical Bessel functions
double cyl_bessel_k(double nu, double x);
… … … … … … … … … … … … … … … … … … … … …
}
```

- Implementation using GSL

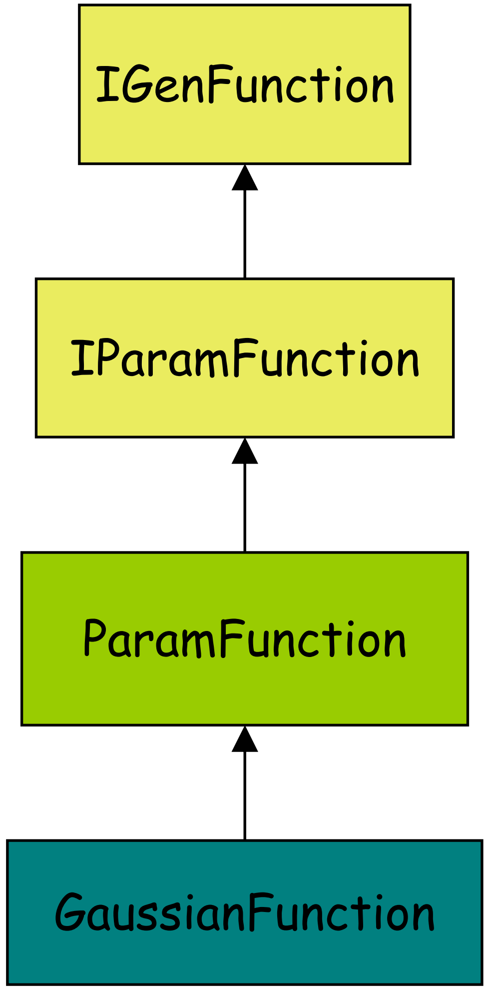
```
#include "gsl/gsl_sf_bessel.h"

// cylindrical Bessel functions (of the first kind)
double mathlib::cyl_bessel_j(double nu, double x) {
  return gsl_sf_bessel_Jnu(nu, x);
}
```

C++ Function classes

- A large variety of use cases (data modeling, plotting) requires additional operations on functions
  - Example: to control the shape of a function will require to access its parameters
- Functions are also used in various numerical algorithms
  - Need to have a coherent signature
  - Use C++ advantages to simplify life to end-user
    - » Have well defined set of interfaces and base classes
- Need for function operations
  - arithmetic operations, composition, convolution
    - » Very useful in the interactive environment

C++ Function Design

- Abstract Generic Function interfaces
  - **IGenFunction** : interface for 1D functions defining :
    - » double operator() (double x)
    - » double gradient(double x);
  - **IMDGenFunction** : multidimensional interface
    - » double operator() (const std::vector<double> & x)
- Abstract interface for parametric functions
  - **IParamFunction, IMDParamFunction**
  - 1D and multi-dim parametric functions:
    - » Defines set/getParameters()
    - » parameterGradient()
- Base Parametric function class
  - User convenient class containing default implementations
  - Not abstract, so derived classes do not need to re-implement all methods
- Sets of concrete classes implementing pre-defined functions:
  - Gaussian, Exponential, Polynomial



Numerical Algorithms Design

- Algorithms API will be based on abstract functions but also on a generic template function.
  - maximum flexibility, user can pass either
    - » an instance implementing an abstract function
    - » an instance of any object implementing some pre-defined operations: operator() , gradient() , etc..
  - Using a concrete function class would avoid virtual function calls when evaluating the function inside the algorithm
- Separate API classes from implementation
  - Hide algorithm implementations
- Have interfaces defining complex algorithm (Minimizer interface)
- Algorithms can be loaded dynamically as plug-in's
  - design an algorithm interface (e.g. Minimizer interface)

Example: Numerical Integration

- Integrator class
  - implemented as wrapper to GSL integration routines
  - Specify type of integration algorithms and integration rule in constructors using an enumeration
  - have also a method directly passing C function pointers (same signatures required by GSL) to avoid adapters

```
Class Integrator {
// constructors
Integrator( const Type type=ADAPTIVE, const Rule rule = GAUSS31, double absTol = 1.E-9, double relTol =
1E-6, size_t size = 1000);
…..
// generic integration method
template < class Function >
double integrate ( const Function & f, double a, double b);

// specialization for GSL function signature ( no adapter needed)
typedef (* GSLFuncPointer) (double , void * );
double integrate ( const GSLFuncPointer & f, double a, double b);
… …
};
```

Example: Root Finder

- Have root finding algorithms for function with and without derivatives (Bisection, Newton, etc..)
- Root Finder class
  - Template class with algorithm type as parameter
  - Static check if function is compatible with algorithm

```
class MyFunction {
public:
 double operator() (double x) { return x*x - 5; }
};

MyFunction f;
// initialize a Root Finder based on Bisection Algorithms
mathlib::RootFinder<mathlib::RootFinder::Bisection> r;
r.setFunction<MyFunction>( function, 0, 5);
r.solve( );
double solution = r.root();
```

Tests and Validation studies

- Performed extensive evaluation of GSL
  - Test numerical quality and performance of special functions
    - » Comparison with ROOT and NagC
    - » Good results obtained by GSL for most used functions
- Random number generator tests
  - New tests designed for correlations and to detect non-random sequences
- Linear Algebra performance tests
  - Comparison of various packages (CLHEP, GSL, BLAS/LAPACK, ROOT, UBLAS) in matrix operations used in track fitting
- See previous talk of M. Hatlo at AAM on September 22, 2004

Fitting and Minimization

- Completed major developments of C++ MINUIT
  - Reached same functionality as in Fortran version
  - Performed validation tests
    - » Same numerical accuracy and CPU performances
  - See Matthias Winkler's presentation in June 16 AAM
  - Now extending it, adding a new minimizer algorithm (FUMILI)
- MINUIT C++ is used by CMS reconstruction and end-users
  - Stand-alone package which can be built independently
- Provided a Fitting library (FML) for standard fitting problems based on MINUIT
  - User convenient package on top of MINUIT for fitting
  - Have also Python interface for interactive users (PyFML)

Status of C++ MINUIT

- MINUIT has been completely re-written in C++
- Not just Fortran -> C++ translation
  - Based on a OO design
  - Set of different classes each performing a well defined task
- Developments are almost complete
- We have same functionality present in the Fortran version:
  - Minimizers:
    - » Migrad, Simplex, Minimize, Scan
  - Error analysis:
    - » Hesse, Minos and Contours
  - Control of Parameters :
    - » fix, set/ remove limits on single and double side
    - » single side limits are NEW, were not in the Fortran version

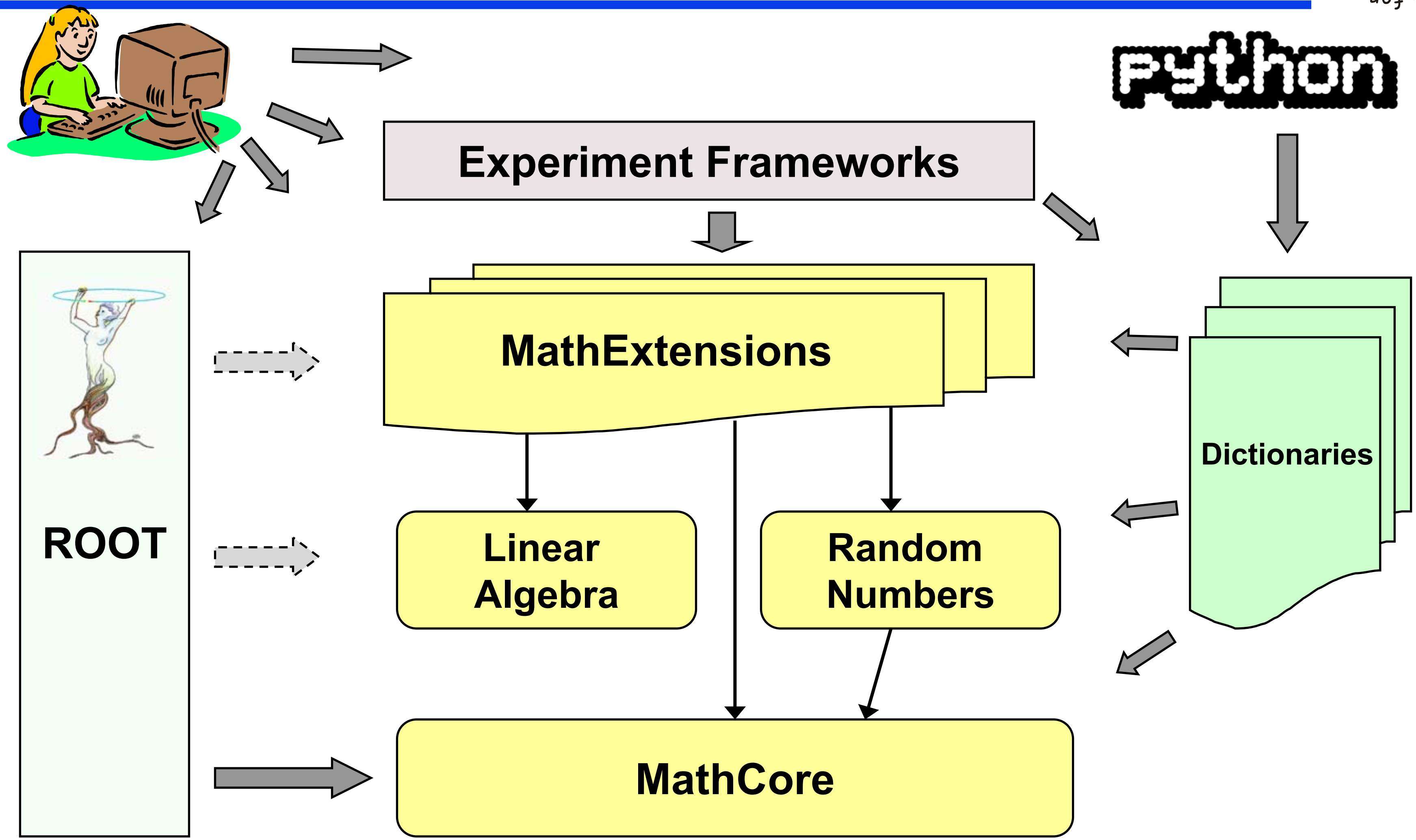
Evaluation of C++ MINUIT

- Extensive tests performed comparing with Fortran and ROOT version
- Results are very satisfactory
  - Same numerical results
  - Same function calls
  - Small penalty observed only for easy functions
    - » 10% for $y=x^2$, slightly more for multidimensional functions
  - no difference for computational expensive functions
- Easy to integrate in external packages
  - interface to ROOT exists
- Used already in CMS reconstruction code

Fitting and Minimization (FML)

- Package for fitting and minimization
- Solve standard fitting problems
  - Chi2, Likelihood (binned and un-binned) fits
  - Provides set of pre-defined model functions
    - Gaussian, Exponential, Polynomial, etc…
  - Support also for user defined functions
- Defines interfaces for minimization
  - Current implementation uses MINUIT
- Very efficient in terms of performances
- User convenient package on top of MINUIT
- Latest release contains also dictionary library for API classes
  - PyFML package for fitting from Python using MINUIT

MathLib usage



Summary

- Providing support in Math Libs for LHC experiments
  - Inventory of functions and algorithms available online
- Design for C++ mathematical libraries
  - Good collaboration with ROOT
  - We are taking into account its requirements
- Produced a prototype implementation of core library (special functions + core algorithms)
  - First version will be available in next SEAL release (milestone done)
  - Test now integration with ROOT
- Performed validation tests of GSL
  - Have confirmed the numerical quality of the library
- Delivered C++ MINUIT with same functionality as in Fortran
  - Completed with a fitting library (FML) and python bindings
  - We are extending it adding FUMILI minimizer
- Starting providing libraries to experiments
  - Trying to involve users in trying the libraries
  - and we will work on the received feedback

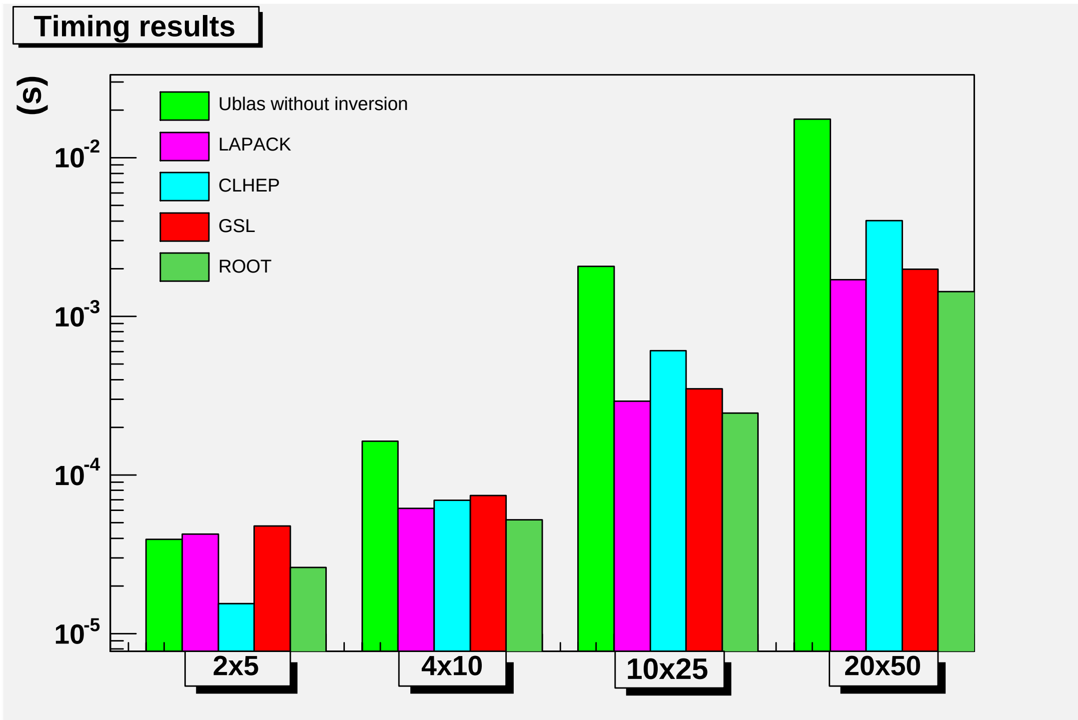
More Information

- Links:
  - MathLib project Web pages:
    - www.cern.ch/mathlib
  - MINUIT pages:
    - www.cern.ch/minuit

    with documentation (User Guide and minimization tutorial)
    and links to download code (can be built easily with configure/make)
- Mailing lists:
  - forum-mathlib@cern.ch
  - forum-minuit@cern.ch

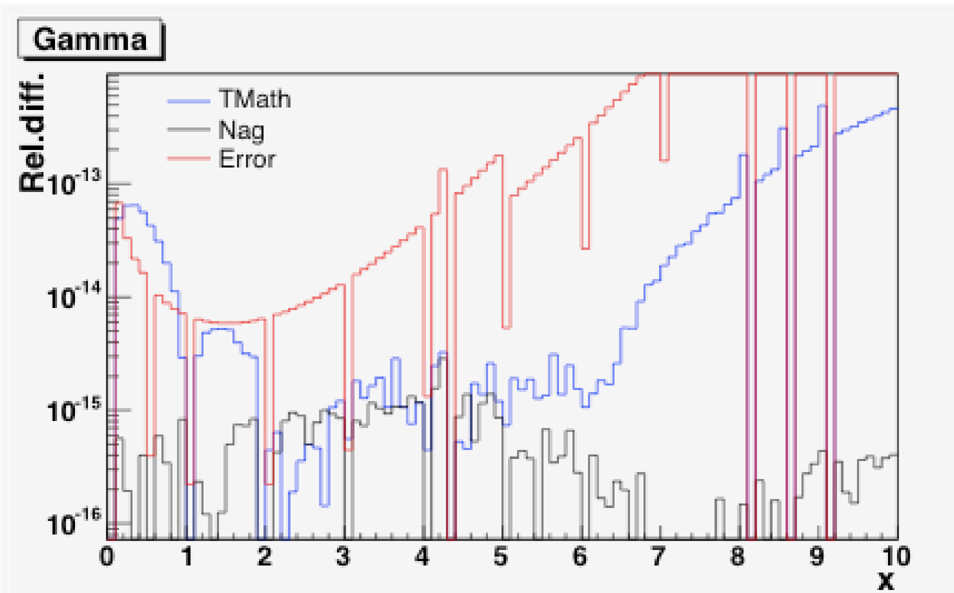
Linear Algebra studies

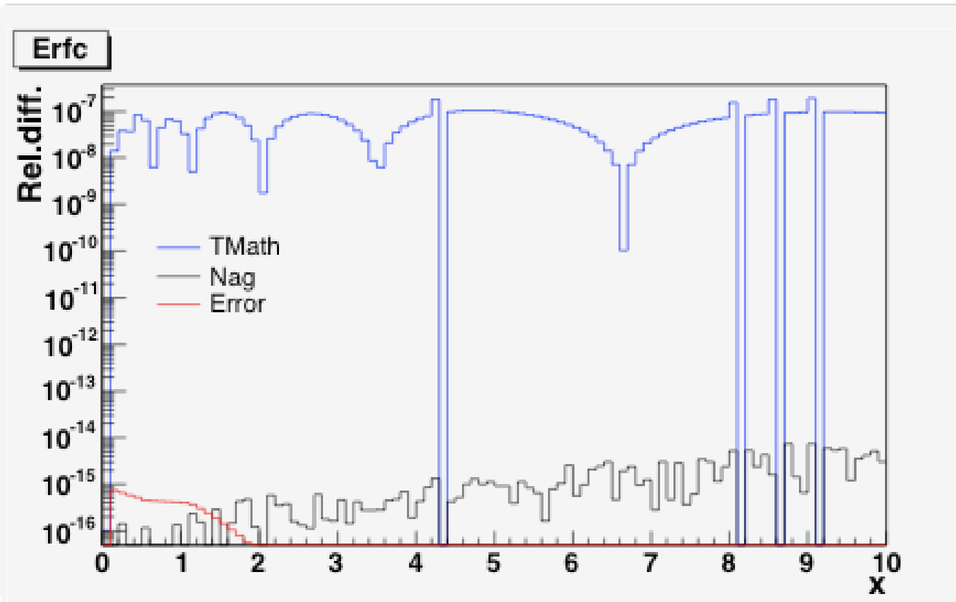
- Measure time spent in operations used in Kalman filter (track state update)
  - Involve multiplication, matrix inversion and transpose
- Compare UBLAS (Boost), BLAS/LAPACK, CLHEP, GSL, ROOT (v. 4)



Evaluation of existing libraries (GSL)

- Test numerical accuracy and time performances of GSL, NAG and ROOT
  - Compare special functions (Bessel, Gamma, Erf) and some statistical functions (e.g. Chi2 probability)
- Good numerical results obtained from GSL





Tests of random number generators

- Study a palette of generators from GSL
- Apply tests looking for correlation and defects in the random sequence
  - Look for some frequency for correlated effects
  - Look at distances between sequence of points
- All generators considered passed the tests