Status report on Math Library activities

Application Area Meeting,
June, 16th 2004

Lorenzo Moneta
CERN-PH

Math Library Project Goals

- The purpose is to provide a coherent Mathematical Library to the end-users
  - The goal is to share a common mathematical library between ROOT and the rest of LCG activities and experiments
- A major requirement is to use the same basic Mathematical Library in all environments
  - Directly in C++ within the experiment reconstruction or simulation programs
  - During the analysis phases from an interactive environment, using either Python or ROOT/CINT

Math Libraries working items

- MathLib WebSite
  - Produce inventory of most common used Mathematical functions and algorithms
- GSL Evaluation
- Development of a C++ MathLib
- Linear Algebra
  - Evaluate and study existing packages
- CLHEP support
  - participation in maintenance and provide consultancy
- Fitting and Minimization
  - C++ Minuit

Mathematical Inventory

- Inventory of common Mathematical Functions and Algorithms needed by the HEP community
- First version is available from MathLib Web page
  - http://seal.web.cern.ch/seal/snapshot/work-packages/mathlibs/mathTable.html
- Starting point is CERNLIB set
- Follow GSL organization (most complete provider)
  - Categorize them according to functionality
- Inputs also from CLHEP and ROOT
- Link each entry with documentations from GSL, CERNLIB and ROOT

Mathematical Inventory (2)

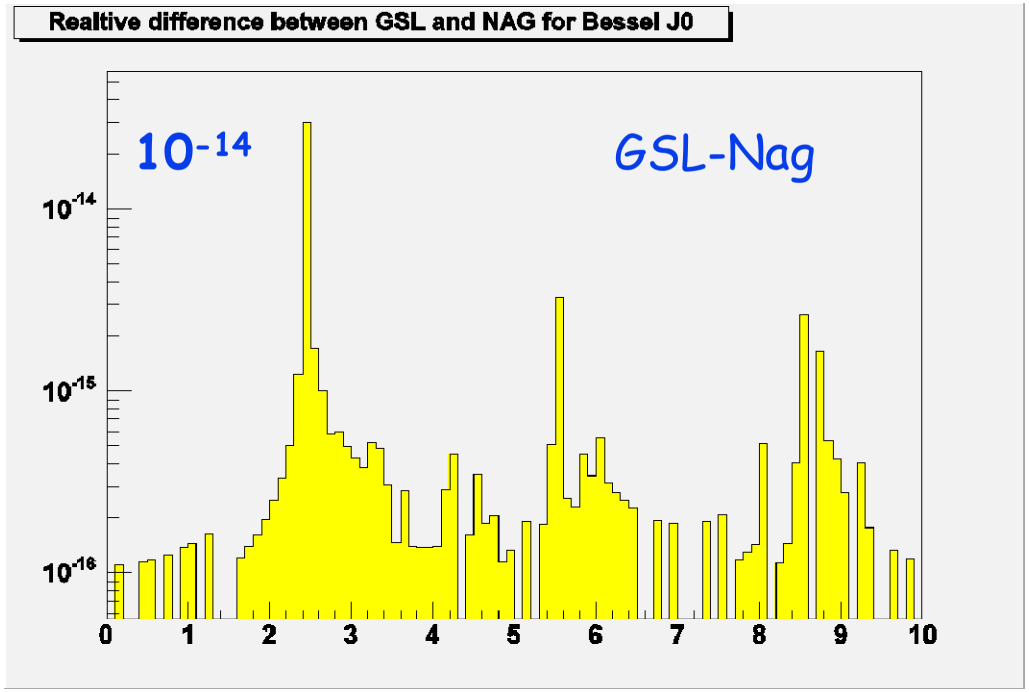
- To be done :
  - Produce test for each entry and add a link to test results
    - » e.g. results from GSL evaluation studies
  - Give recommendation on the various implementation
  - Add link to MathLib C++ API documentation
    - » How to use this function in the MathLib library
  - Produce links from CERNLIB list to new MathLib list
  - Produce FAQ list

GSL Evaluation

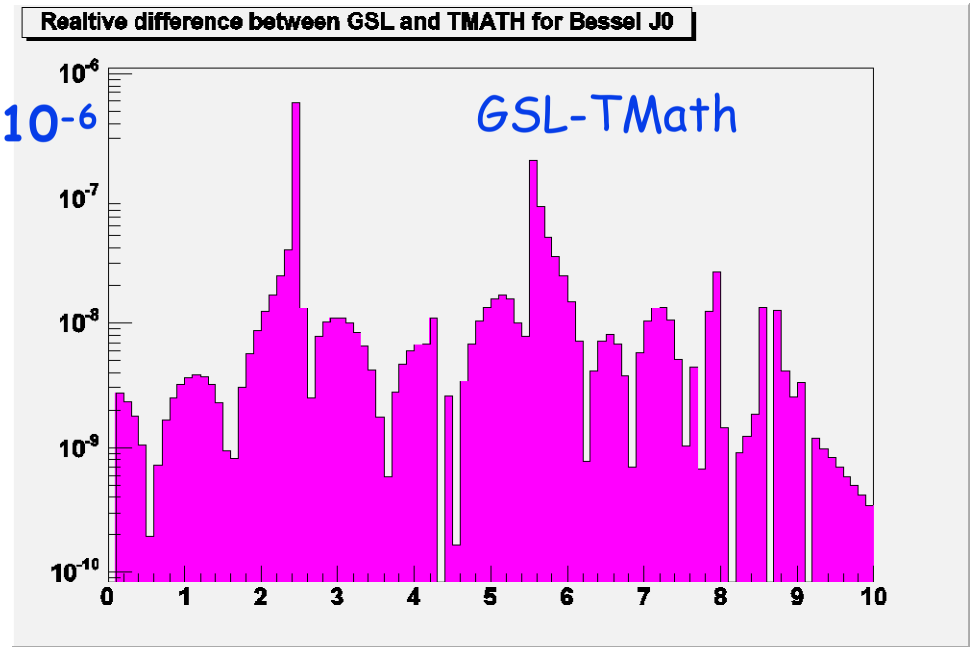
- Evaluate GSL studying numerical quality and performances
  - Compare execution time
- Started from special functions and pdf
  - Compare numerical result with ROOT (TMath) and Nag
  - Study numerical results and measure CPU time
- Test studies of random number generator
  - Look for defects in the generator in particular in correlated effects
- Priority is in what is used by experiments
  - test extrapolation, root finders, integration methods
- Goal is to build automatic test suite to be run automatically for every new GSL release
- Tests will be linked to MathLib Web pages

Numerical studies of GSL special functions

- Look at the relative difference GSL/Nag and GSL/Root

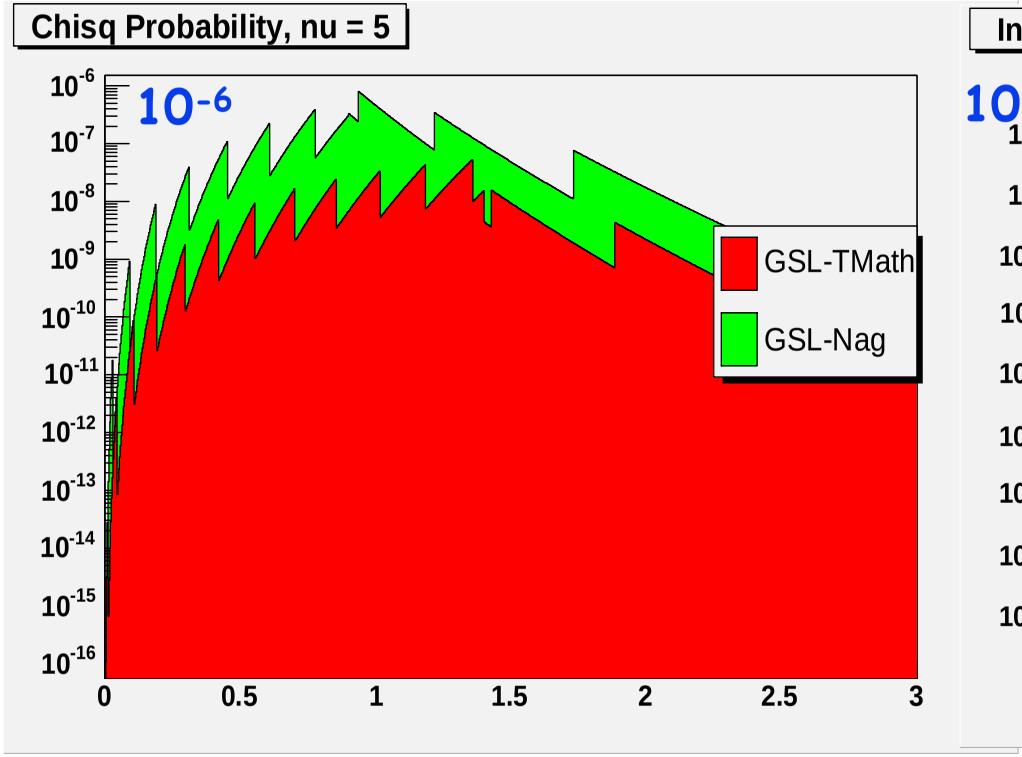


Note the difference in the scale

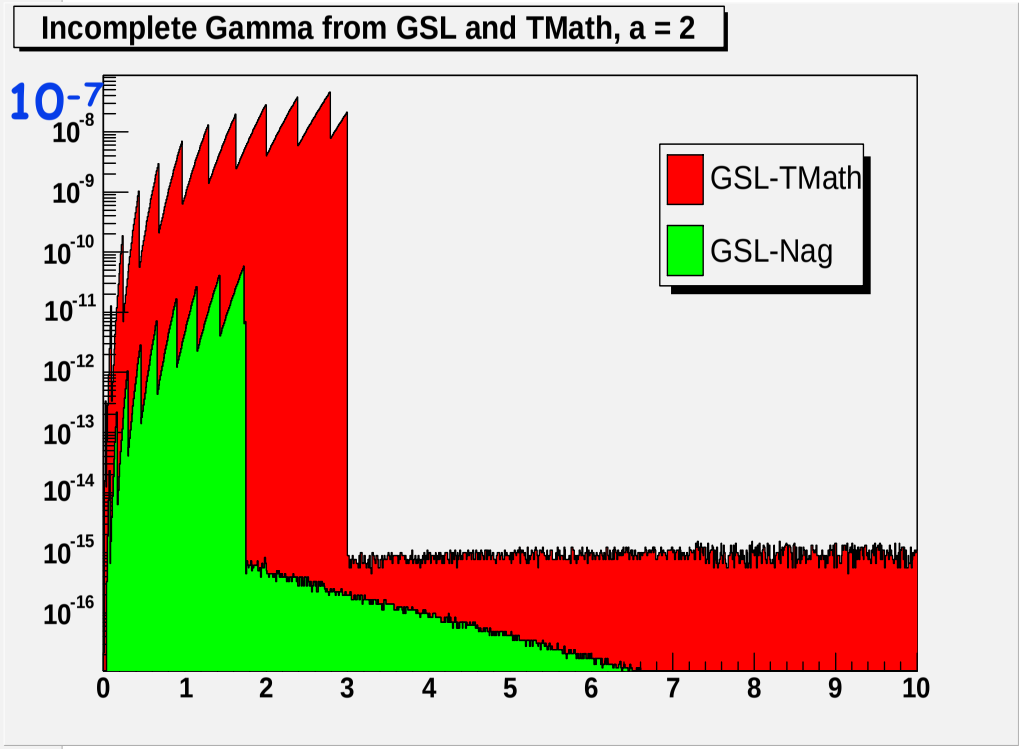


GSL Function studies

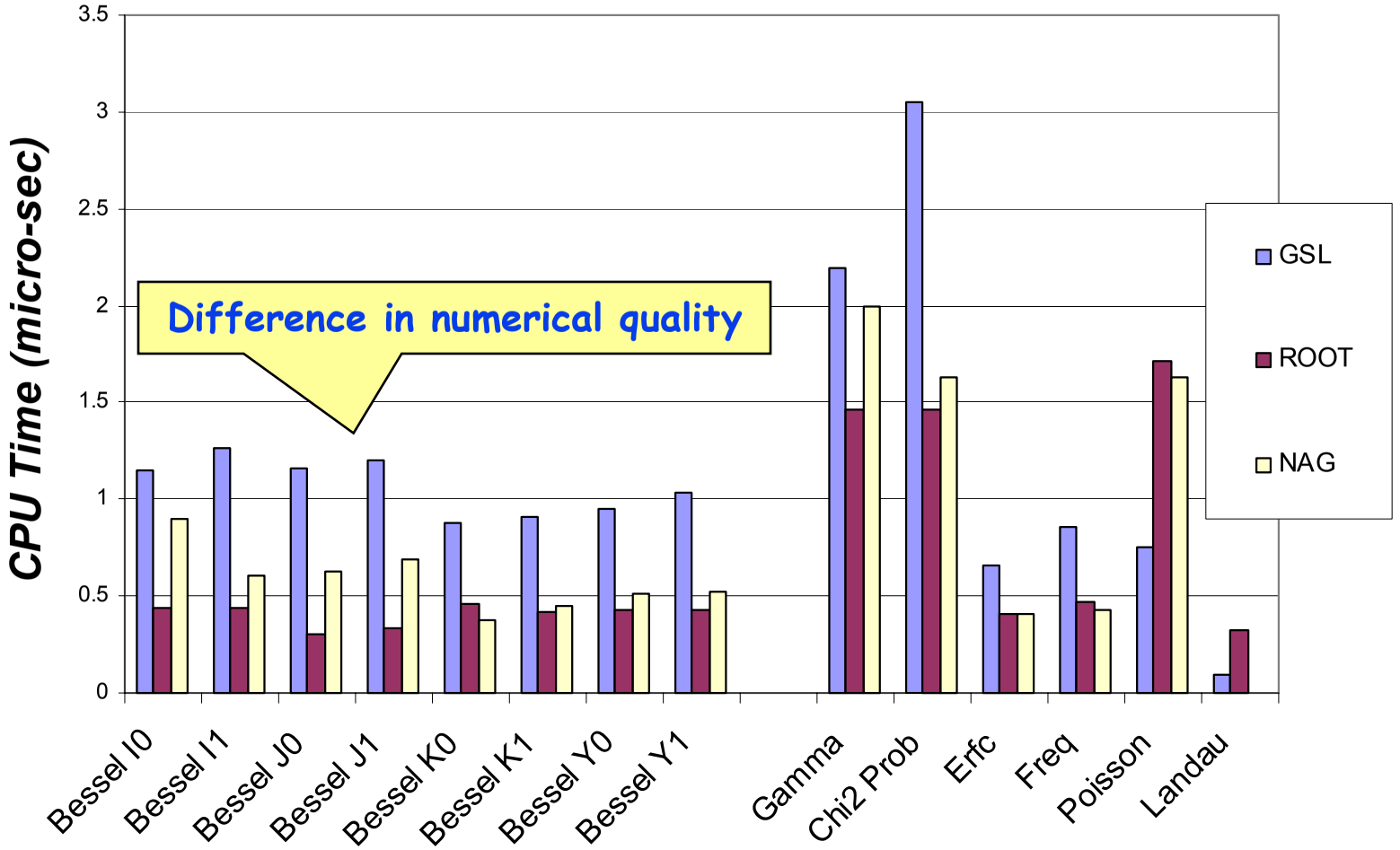
χ^2 Probability function



Gamma function



Performance Tests

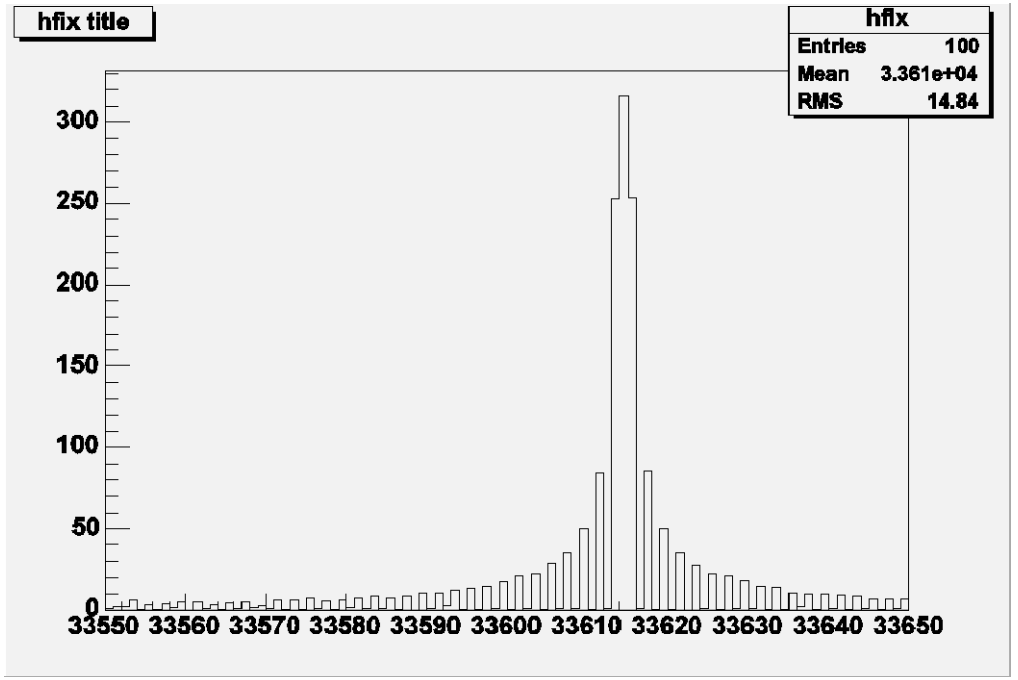


Random number generators tests

- Look at defects:
  - In a selected region (i.e 0< x < 0.1) bin the data in N_{bin}
  - Look at difference in :

$$S = \frac{\sum_{evenbins} N_i - \sum_{oddbins} N_i}{\sqrt{N_{Tot}}}$$

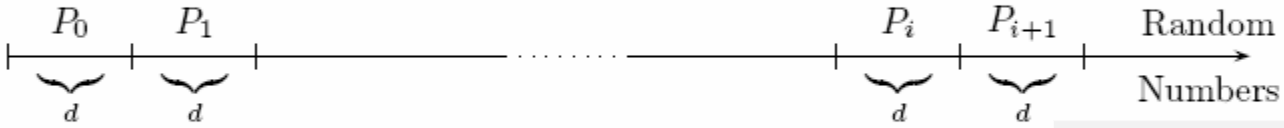
  - Select value of N_{bin} (frequency) where a peak in S is observed
  - Look in other regions for the same frequency if a pattern is observed



Frequency found for Park and Miller generator (gsl_rng_minstd)

Random number generators tests (2)

We produce a sequence of random numbers, $x_0, x_1, x_2, ...$
For dimension d we have the points $P_0 = (x_0, x_1, ..., x_{d-1})$
and $P_1 = (x_d, x_{d+1}, ..., x_{2d-1})$.
We then look for a point $P_i = (x_i, x_{i+1}, ..., x_{i+d-1})$ which is close to P_0,
when $|P_i - P_0|$ is small.
When we find such a point we check whether $|P_{i+1} - P_1|$ is small as well.

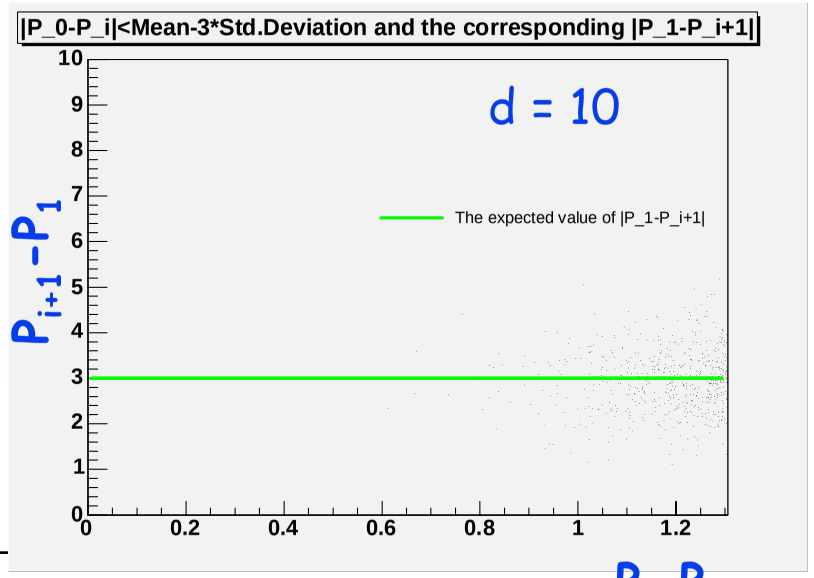


For the distance we use the following measure

$$|P_i - P_0| = \sum_{k=0}^{d-1} |x_{i+k} - x_k|$$

We say that two points are close if

$$|P_i - P_0| < \mu - 3\sigma$$

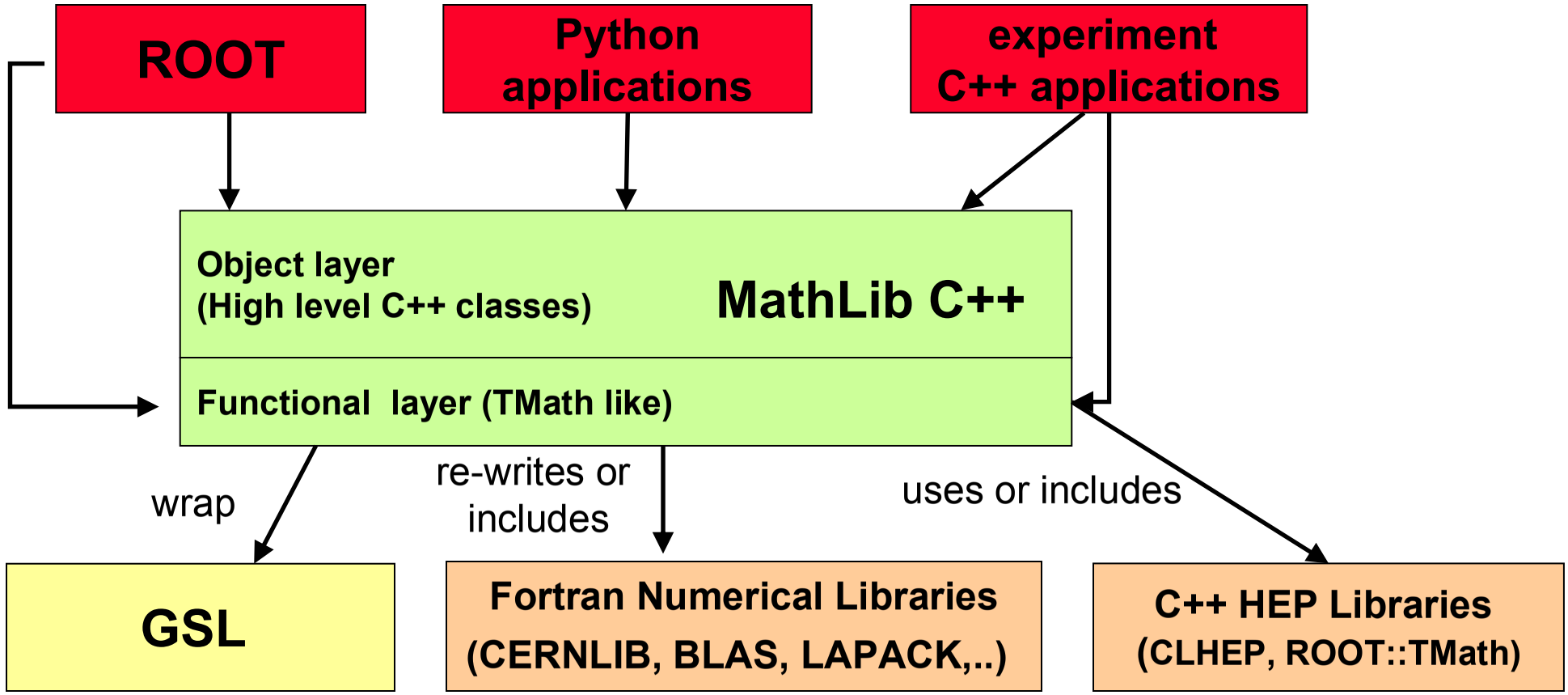


Major generators in GSL pass the test

C++ MathLib

- Functional C++ layer for functions and algorithms
  - A wrapper layer to hide implementation
  - Use or copy in the MathLib repository code from external providers
    - GSL, ROOT TMath, CERNLIB, BLAS, LAPACK, etc..
  - Decision to use or copy will depend on the amount of code needed
    - » GSL, since will provide a large part of functionality, will be probably used
    - » Rewrite/copy what's missing from CERNLIB (e.g. Minuit)
- Higher C++ layer containing:
  - Function classes for minimizations, integration, extrapolation,...
  - Matrices classes for linear algebra operations, etc...

C++ MathLib


ROOT
Python applications
experiment C++ applications
Object layer (High level C++ classes)
MathLib C++
Functional layer (TMath like)
wrap
re-writes or includes
uses or includes
GSL
Fortran Numerical Libraries (CERNLIB, BLAS, LAPACK,..)
C++ HEP Libraries (CLHEP, ROOT::TMath)

C++ functional layer

- Functional C++ wrapper layer to functions and algorithms
  - Make them free functions in a namespace
    - » Allow user to extend them adding additional functions
  - Approach adopted by C++ standard committee
  - Use name scheme as in C++ standard proposal

```
namespace mathlib {
        double bessel_I(double l, double x);
        double bessel_J(double l, double x);
        ….
        double erfc (double x);
}
```

- Start wrapping GSL which provides the vast majority of the needed functions

C++ Object Layer

- Various advantages for having Math C++ classes
  - user convenience, genericity (templates), etc..
- Functions classes describing parametric functions are probably needed
  - Function with a state (their parameters)
  - Various algorithms need functions :
    - » Function fitting and minimization, interpolation, integration, etc…
  - Persistency needs parametric function classes
- Generic interface for users to plug in easily new functions in the algorithm
- Higher level analysis tools and frameworks could use this interface and extend it according to their needs
- Design together with expert users and provide prototype for rapid evaluation

Matrix classes

- Matrix and Vector classes needed
  - Use power of overloading C++ operator for matrix operations
  - These objects belong to event model and need to be made persistent
- Using expression templates for matrix operations
  - Used by Minuit
  - Written C++ Wrapper to Fortran BLAS/LAPACK and GSL
    - » need to be extended and optimize memory allocation in case of small matrices

Linear Algebra Studies

- Developed a C++ wrapper layer to BLAS/LAPACK
  - Using expression templates for matrix operation
- Comparison of various linear algebra packages
  - Study performance in evaluating Kalman filter equations for update of track state (from E. Myklebust – see report)
    - » Operations involving small matrices (up to size 5)
    - » Comparison of CLHEP, BLAS/LAPACK, GSL and uBLAS
    - » CLHEP best performer
      - dedicated matrix inversion algorithm
      - Uses stack allocator up to size=5
    - » BLAS/LAPACK wrapper x1-2 slower but uses heap allocation
    - » GSL 1.3-2x slower than BLAS/LAPACK
    - » Boost uBLAS slowest of all (unclear, needs further investigation)
  - Plan to compare also with new ROOT Linear Algebra and other packages (tvmet, MET, etc…)

Track fitting

- Kalman filter update of a track state. Measure time in the various steps

Results

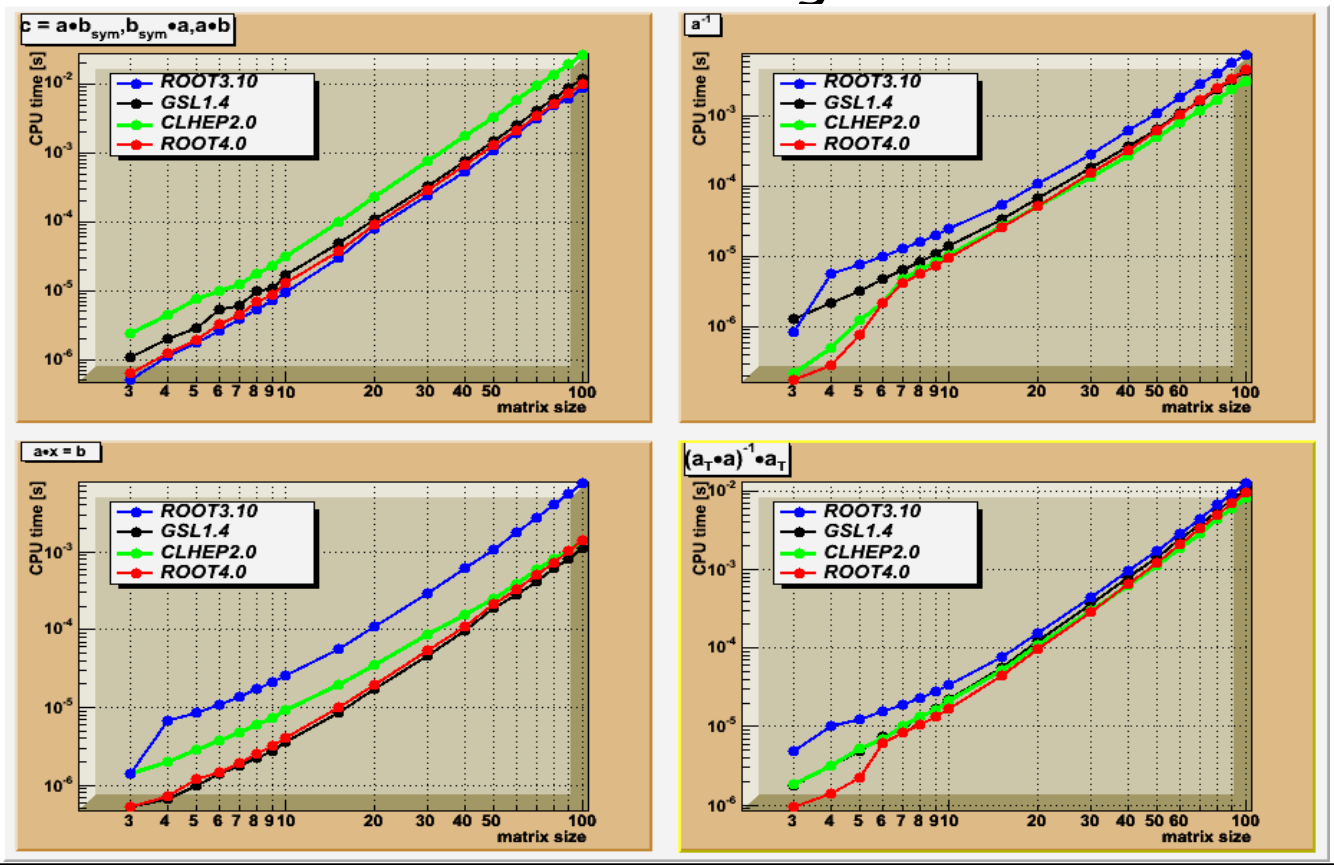
1...update of state vector x
2...calculation of Kalman gain matrix K
3...update of state error C
4...calculation of local Chi2 increment (not part of Kalman filter step)

numbers are in "clock_ticks (relative_to_CLHEP)"

	CLHEP	BLAS/L.	GSL	uBLAS, heap	uBLAS, stack
1	2,334	3,117 (1.34)	6,716 (2.88)	3,342 (1.43)	1,827 (0.78)
2	9,330	14,800 (1.59)	25,950 (2.78)	136,900 (14.67)	144,100 (15.44)
3	6,223	6,092 (0.98)	8,395 (1.35)	15,920 (2.56)	15,090 (2.42)
4	5,893	12,500 (2.12)	20,000 (3.39)	skipped	skipped
1+2+3	18,760	25,090 (1.34)	46,670 (2.49)	163,300 (8.70)	189,000 (9.20)

Linear Algebra Studies (2)

- Comparison studies from E. Offermann
- Compare new ROOT 4.0 Linear Algebra with GSL and CLHEP



Fitting and Minimization

- Minuit
  - New Minuit C++ is almost complete
  - Reached same functionality, quality and performances as in the Fortran version
- Expected new developments:
  - adding specialized minimization algorithms for fitting problems (e.g. chi2 fits) for performance increase
    - » Fumili
  - inclusion should be easy thanks to good OO design
    - » Only few classes needs to be touched
  - Support for linear constraints
- Test also comparing with other minimizers from GSL, NAG, etc…

Fitting and Minimization Framework

- A new prototype package is available in the next SEAL release (1.4.0)
- Fitter interface for standard fitting problems
  - Set of predifined model (fitting) functions
    - » Gauss, exponential, polynomials, etc..
  - and fitting methods
    - » Chi2, Binned and Unbinned Maximum Likelihood
  - Support for user defined functions and methods
  - Minimizer interface
    - » Contains Minuit C++ implementation
    - » Exists also implementation based on Nag or Fortran Minuit
  - Possibility to change minimizer at run time
- Need to complement with Python bindings (and CINT) for interactivity

Resources

- from CERN-SFT :
  - M. P. Hatlo, F. James, L.M., A. Pfeiffer (CLHEP), M. Winkler
- Collaboration from experiments
  - M.Paterno, W. Brown (CMS Fermilab)
- Collaboration with ROOT team and other external project
  - Contributions from Paul Kuntz (HippoDraw) for Minuit
- Project is open for participation of any interested contributors
  - Welcomed contributions from experts and users of Mathematical libraries from the LHC experiments

MathLib Planning

- Project Milestones (defined in SEAL work plan) :
  - MathLib project Web (15/7/2004)
  - First version of C++ MathLib (30/9/2004)
- MathLib Web Site
  - http://seal.web.cern.ch/seal/snapshot/work-packages/mathlibs/index.html
  - Entry point for users (with links to documentation)
    - » Inventory of Mathematical functions and Algorithms
  - Minuit Web pages.
- Mailing lists :
  - Forum-mathlib@cern.ch with WEB archive
- MathLib Meetings (approx every month) :
  - discuss new developments
  - collect requests and feedback from experiments

Summary

- Web Site with list of needed mathematical functions and algorithms already exists (link)
- Evaluation of the GNU Scientific Library is well advanced
  - Various tests have been written
  - numerical quality of GSL looks to be rather good
- Started providing common mathematical libraries for LHC experiments:
  - Minuit C++ almost completed
  - Released fitting and minimization prototype
  - First version of C++ interfaces to functions will be released next
- Project is a good collaboration between LCG, experiments and ROOT
  - Good Participation in the MathLib meetings
  - Try now to involve some users in testing