

MathLib starting meeting

Lorenzo Moneta
CERN-PH

Scope and goals

- ◆ Idea is to have a discussion forum on numerical mathematical methods (mathematical libraries)
- ◆ Active participation from experts and users of Mathematical libraries from the LHC experiments and any other interested users
 - Analysis tools developers (e.g. ROOT), ...
- ◆ Have regular meeting (every 2-3 weeks) with goals to
 - Collect requirements from the experiments
 - Review and discuss current activities
 - Define future work items

MathLibs Plans

- ◆ The purpose is to provide a coherent Mathematical Library to the end-users
 - Coordinate the activities with ROOT, bringing the needs of LCG and ROOT together, trying to avoid maintenance and support of various mathematical libraries providing similar functionality
 - 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
- ◆ Development done in collaboration with the LHC experiments

Initial MathLib working items

- ◆ Following approved SEAL plan:
 - Produce inventory of most common used Mathematical functions and algorithms
 - Evaluation of GSL
 - Develop a C++ MathLib
 - Linear Algebra
 - Fitting and Minimization
- ◆ Expect a discussion for each item

Organizational Issues

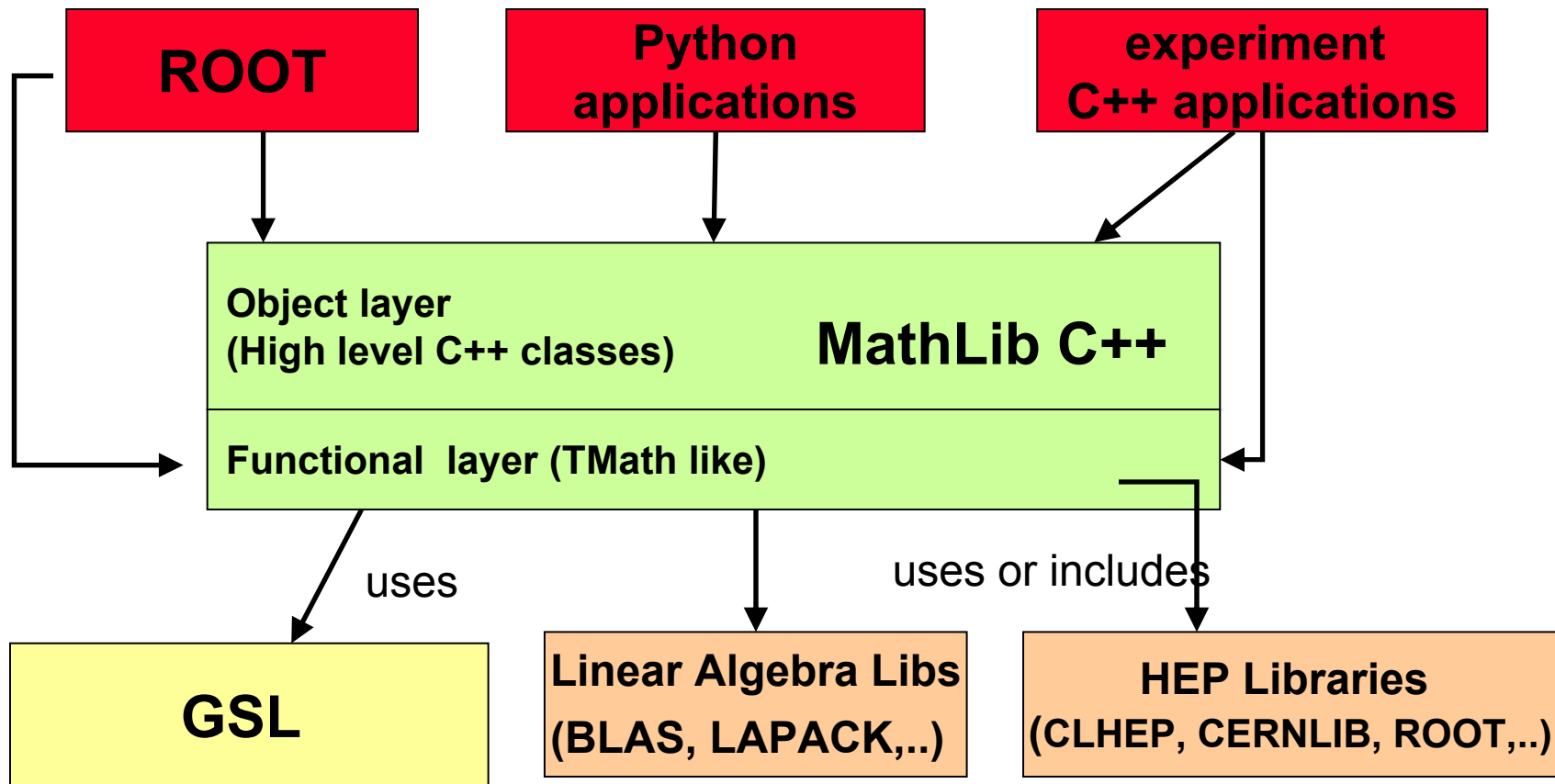
- ◆ Project Milestones (defined in SEAL work plan) :
 - MathLib project Web (15/7/2004)
 - First version of C++ MathLib (30/9/2004)
- ◆ Need to write a detailed technical report
- ◆ MathLib Web page
 - <http://seal.web.cern.ch/seal/snapshot/work-packages/mathlibs/index.html>
 - Entry point for users (with links to documentation)
 - and developers with links to meetings presentations, working documents, etc...
 - Links to code reference documentation
- ◆ Mailing list :
 - Forum-mathlib@cern.ch
 - Need for a developer restricted list ?

Inventory of functions and algorithms

- ◆ Web pages with list of common Mathematical Functions needed by the HEP community
- ◆ First version available from MathLib Web page
 - <http://seal.web.cern.ch/seal/snapshot/work-packages/mathlibs/mathTable.html>
- ◆ Starting point is CERNLIB set
 - good starting point is also list from Walter and Marc
- ◆ Follow GSL organization (main provider)
 - Categorize them according to functionality
- ◆ Need to document each entry and add corresponding references
- ◆ Expect Feedback and comments

C++ MathLib

- ◆ Idea is to have a two layer library



C++ functional layer

- ◆ Functional C++ wrapper layer to functions and algorithms
- ◆ For special functions:
 - Make them free functions in a namespace
 - » Allow user to extend them adding additional functions
 - Approach adopted by C++ standard committee
- ◆ GSL provides the needed functions
 - Wrap GSL function in new function name with appropriate namespace
 - Use name scheme as in C++ standard proposal
- ◆ Open issue is packaging :
 - Copy in GSL code or keep external dependency

C++ function classes

- ◆ Generic 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
- ◆ Design a common interface with minimal methods:
 - `value()`, `get/set parameters()`, `gradient()`
- ◆ Higher level analysis tools and frameworks could use this interface and extend it according to their needs

What already exist ?

◆ Generic functions from CLHEP

- AbsFunction interface
- Implement operation on functions ($+$, $-$, $*$, $/$), composition and convolution

◆ AIDA Function interface

- Pure Abstract interface
- Contain some unnecessary methods (e.g. `annotation()`)

◆ ROOT Function (TF1)

- Contain many non-mathematical methods (related to the ROOT framework and graphics)

CLHEP GenericFunction/AbsFunction

```
class AbsFunction {
public:

    // Constructor
    .....
    // Function value: N-dimensional functions must override these:
    virtual unsigned int dimensionality() const ;           // returns 1;
    // Function value
    virtual double operator() (double argument)             const=0;
    virtual double operator() (const Argument &argument) const=0;
    // Every function must override this:
    AbsFunction * clone() const;
    // Function composition. Do not attempt to override:
    virtual FunctionComposition operator () (const AbsFunction &f) const;
    // Derivative, (All functions) (do not override)
    Derivative derivative(const Variable &v) const;
    // Derivative (1D functions only) (do not override)
    Derivative prime() const;
    // Does this function have an analytic derivative?
    virtual bool hasAnalyticDerivative() const {return false;}
    // Derivative. Overriders may be provided, numerical method by default!
    virtual Derivative partial(unsigned int) const;
};
```

AIDA IFunction

◆ Pure Abstract interfaces to function

```

    virtual double value (const std::vector< double > &x) const=0
    virtual int dimension () const=0
    virtual bool isEqual (const IFunction *f) const=0
    virtual const std::vector<
        double>& gradient (const std::vector< double > &x) const=0
    virtual bool providesGradient () const=0
    virtual std::string variableName (int i) const=0
    virtual const std::vector<
        std::string>& variableNames () const=0
    virtual void setParameters (const std::vector< double > &params)=0
    virtual const std::vector<
        double>& parameters () const=0
    virtual int numberOfParameters () const=0
    virtual const std::vector<
        std::string>& parameterNames () const=0
    virtual bool setParameter (std::string name, double x)=0
    virtual double parameter (std::string name) const=0
    virtual int indexOfParameter (std::string name) const=0
    virtual IAnnotation& annotation ()=0
    virtual const IAnnotation& annotation () const=0
    virtual std::string codeletString () const=0
  
```

ROOT Function

◆ ROOT TF1 contains:

- **Methods strictly related to a function (from TFormula)**
 - » Eval()
 - » GetNDim()
 - » GetParameters() and SetParameters()
 - » GetNPar()
 - » GetParName() and SetParName()
- **Mathematical methods:**
 - » Derivative()
 - » Integral()
 - » GetMaximum(), GetMaximumX(), GetMinimum(), GetMinimumX()
 - » GetRange(), SetRange()

TF1

- ◆ But also much more :
 - Quantity typically related to fit:
 - » Chi2, probability, ndf
 - And plotting
 - » Draw()
 - » setMaximum(), setMinimum()
 - » getXAxis()
 - Random
 - » GetRandom()
- ◆ All useful things but mainly bound to ROOT
- ◆ Some functionality better being delegated to helper classes
- ◆ Difficult to integrate and use in other tools and frameworks

Alternative/complementary approach

- ◆ Functional approach (like in STL)
- ◆ use of functors (e.g. boost function)
- ◆ No need for user to inherit from a an abstract function class
- ◆ To be generalized requires heavy use of templates
- ◆ Example using boost::function
 - Very flexible, can be used with free functions, class member functions, etc..

```
typedef boost::function< double( const std::vector<double> & ) > Func;  
  
void Minimizer::minimize( Func f,      .... ) { ..... }
```

Next steps...

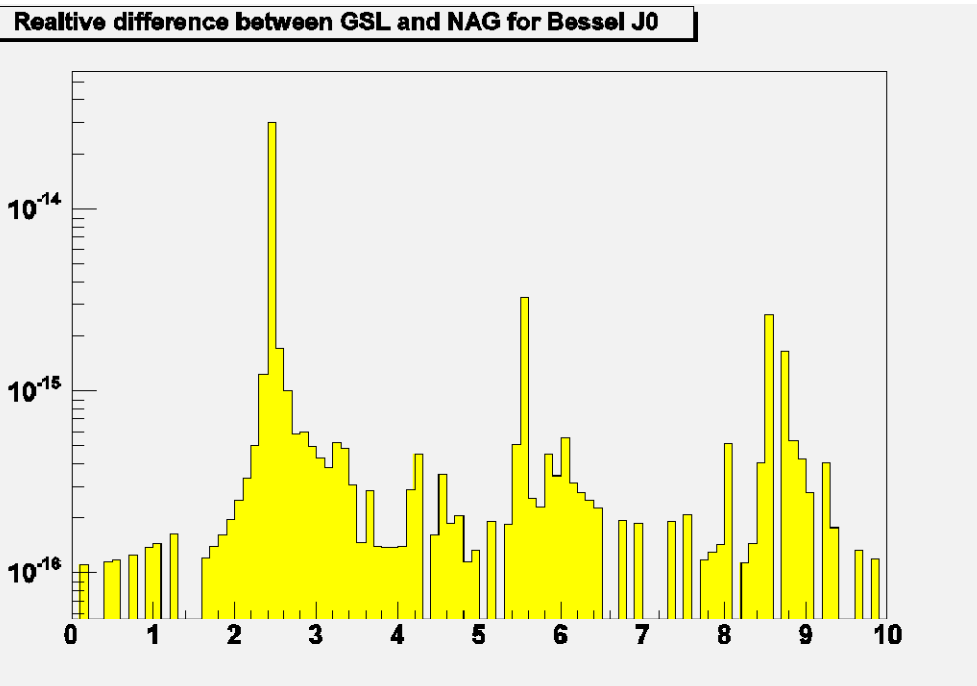
- ◆ Start implementing C++ wrapper layer
 - Start with special functions listed in the C++ standard proposal
- ◆ Discuss design for higher C++ layer
 - Concentrate first on functions
 - Design function classes (interfaces)
 - Look at what already exist (CLHEP, ROOT, AIDA, GSL++, etc...) for functions
- ◆ Come up with a prototype to perform some evaluation studies

GSL Evaluation

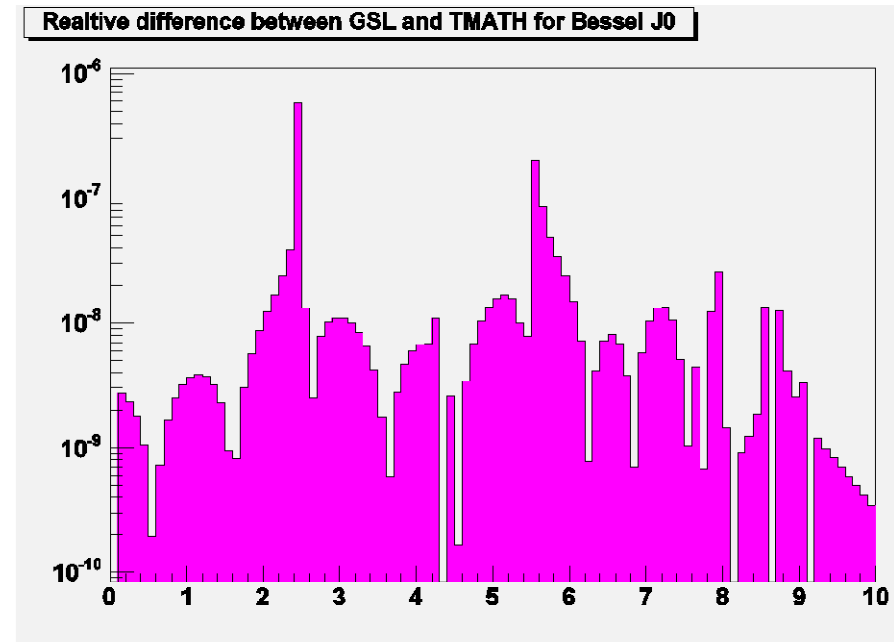
- ◆ Initiated studies here at CERN evaluating some functions of GSL (performed by Marte Hatlo)
- ◆ Studied first GSL special functions:
 - Compare numerical results with Nag and ROOT
 - GSL vs ROOT TMath performances
- ◆ Test studies of random number generator
 - Look for defects in the generator
- ◆ Next step is studying Integration methods
- ◆ Goal is to build a test suite to be run automatically for every new GSL release
- ◆ Any desire to have any part of GSL to be evaluated and tested ?

Numerical studies of GSL special functions

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



Note the difference in the scale



Timing Performance of GSL vs ROOT special functions

◆ CPU time for direct GSL, wrapped GSL and TMath



Tests of Random number generators

◆ Look at defects:

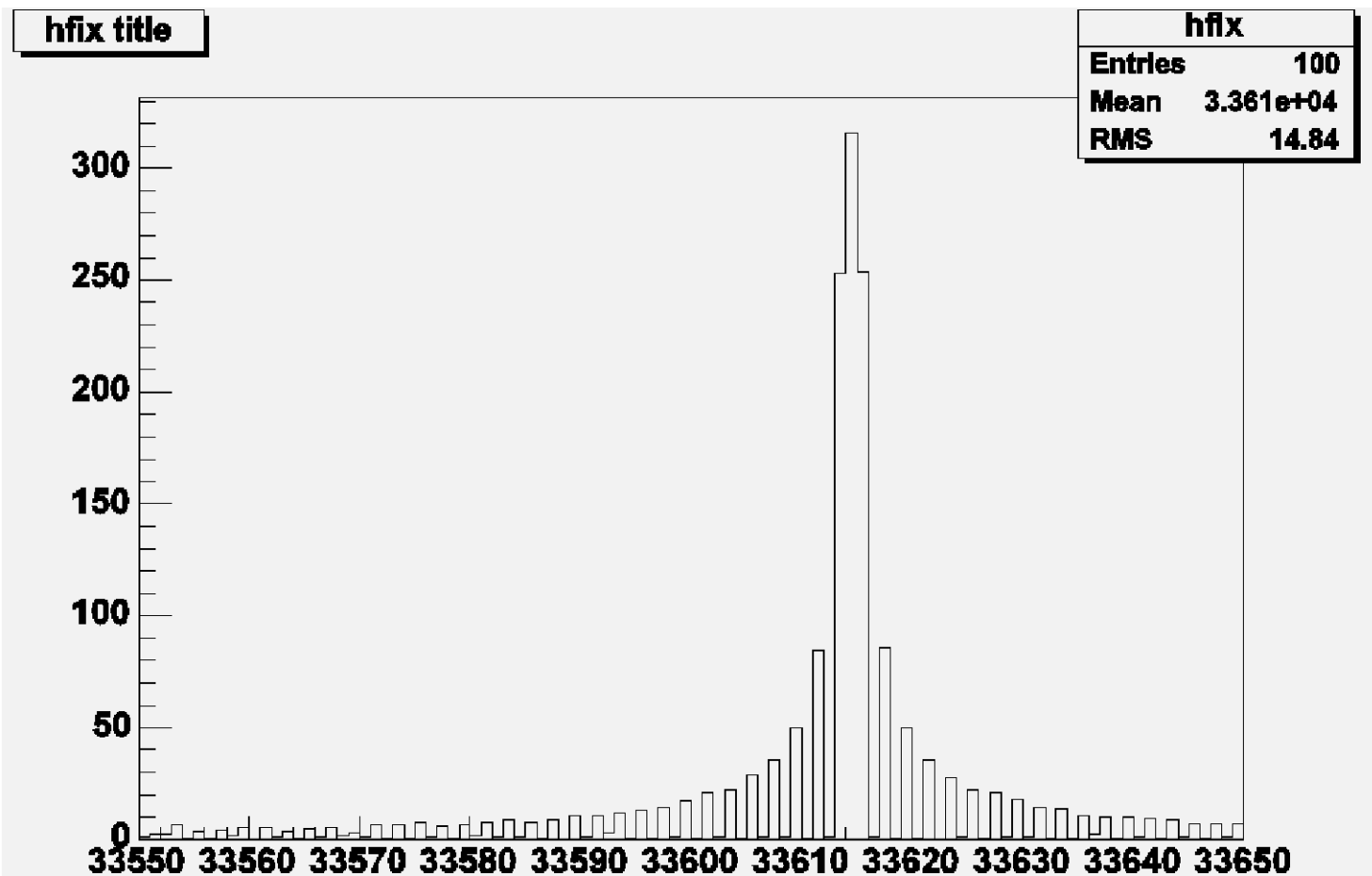
- In a selected region (i.e $0 < x < 0.1$) bin the data in N_{bin}
- Look at difference in :

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

- 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

Random Number Generator tests

- ◆ Frequency found for Park and Miller generator (gsl_rng_minstd)



Linear Algebra Studies

- ◆ Developing a C++ wrapper layer to BLAS/LAPACK
 - Matthias will give a presentation at the following meeting
- ◆ Comparison of various linear algebra packages
 - Summer student's study in the context of track fitting ([PDF doc](#))
 - » inversion and multiplication of small matrices
 - » Comparison of CLHEP, BLAS/LAPACK, GSL and uBLAS
- ◆ Root studies performed by E. Offermann and presented at the ROOT SLAC workshop
- ◆ Not possible to find general best algebra package
 - We should concentrate to use case relevant to us
 - Ex: track fitting: operations on small matrices
 - Large sparse matrices lower priority
- ◆ Recommend optimal solution according to needs

Matrix inversion

◆ CLHEP vs LAPACK

- CLHEP uses stack allocation for sizes ≤ 6

<i>size</i>	<i>LAPACK</i>	<i>CLHEP</i>	<i>LAPACK/CLHEP</i>
2x2	2230	264	12.89
4x4	5000	940	5.83
6x6	8780	2000	4.57
8x8	13220	19530	0.68
10x10	18740	31470	0.59

Table 3. Time (clockticks) for inverting symmetric matrices. The quota of the time for the two libraries are calculated after the timer overhead has been subtracted. Tests run on a Pentium4 1.8 Ghz.

Track fitting

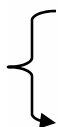
- ◆ Kalman filter update of a track state. Measure time in the various steps

Matrix
version

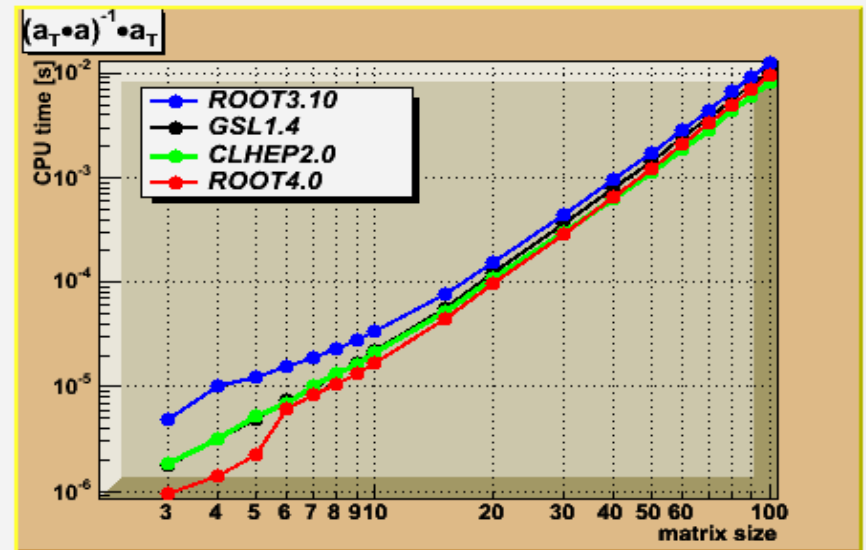
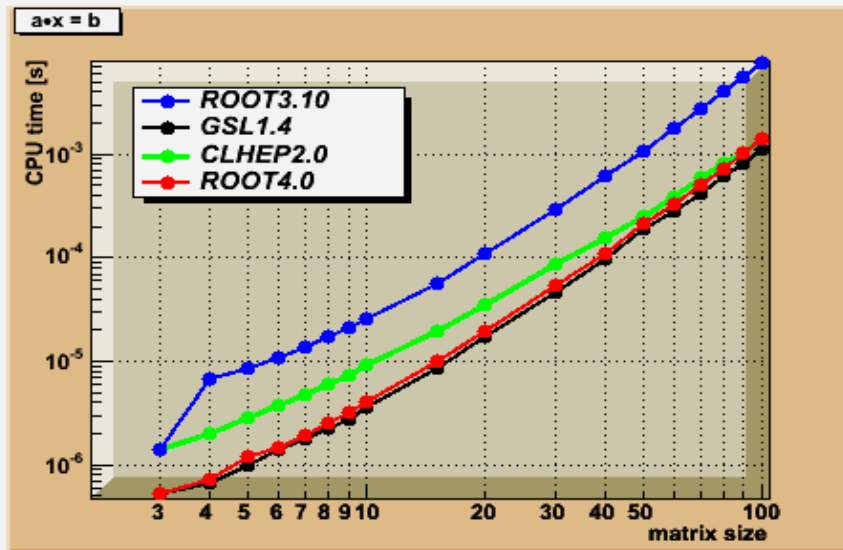
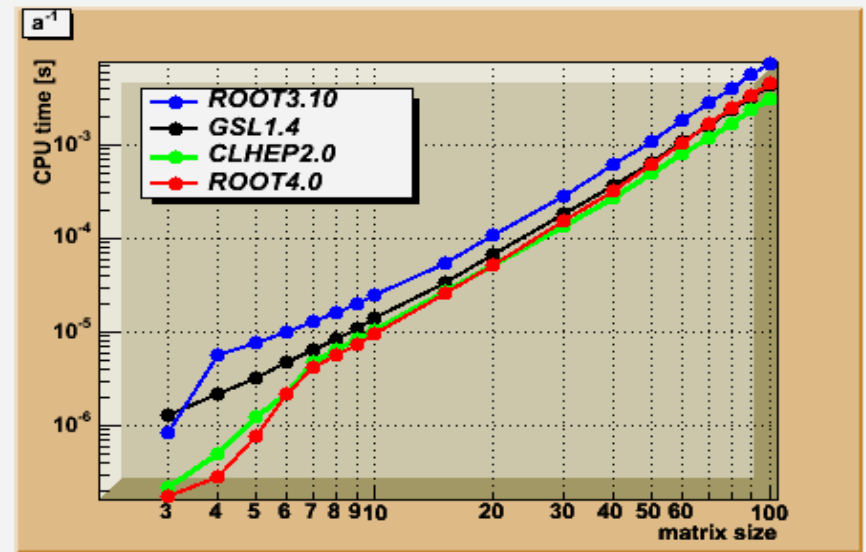
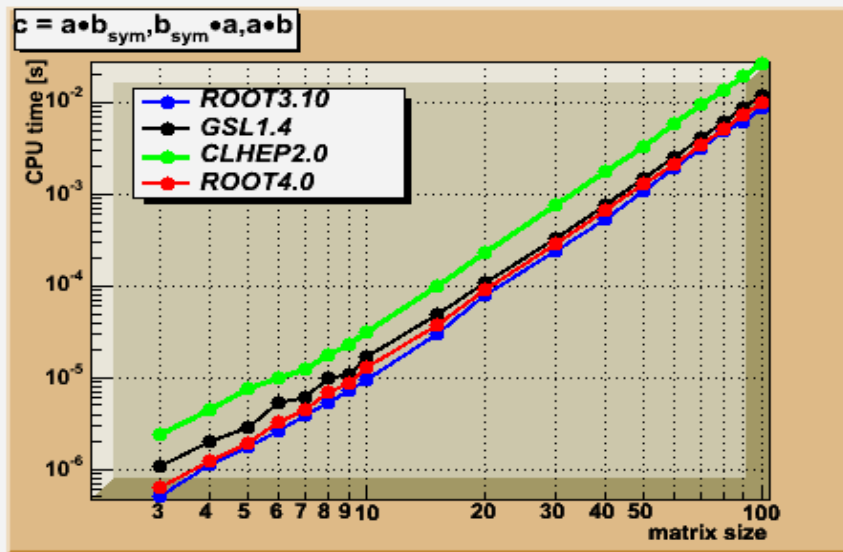


	<i>CLHEP</i>	<i>BLAS/L.</i>	<i>GSL</i>	<i>uBLAS, heap</i>	<i>uBLAS, stack</i>
1	2,334	3,117 (1.34)	6,716 (2.88)	3,342 (1.43)	1,827 (0.78)
2 ¹	9,330	14,800 (1.59)	25,950 (2.78)	136,900 (14.67) ⁶	144,100 (15.44) ⁶
2 ²	8,868	16,330 (1.84)	25,950 (2.93)	21,150 (2.38) ⁶	18,600 (2.10) ⁶
2.1	4,600	4,869 (1.06)	7,462 (1.62)	8,571 (1.86)	7,469 (1.62)
2.2 ³	260	3,794 (14.59)	5,800 (22.31)	⁵	⁵
2.2 ⁴	478	5,112 (10.69)	9,684 (20.26)	⁵	⁵
2.3	4,776	4,895 (1.02)	11,980 (2.51)	10,910 (2.28)	9,980 (2.09)
3a	6,223	6,092 (0.98)	8,395 (1.35)	15,920 (2.56)	15,090 (2.42)
3b	17,010	14,160 (0.83)	21,350 (1.26)	257,100 (15.11)	263,100 (15.47)
4	5,893	12,500 (2.12)	20,000 (3.39)	⁵	⁵
5a	18,760	25,090 (1.34)	46,670 (2.49)	163,300 (8.70) ⁶	189,000 (9.20) ⁶
5b	30,210	33,830 (1.22)	59,140 (1.96)	411,100 (13.61) ⁶	427,600 (14.15) ⁶

Total
time



PCBRUN: Pentium IV 2.4 Ghz 512 Mbytes RAM, IDE disk



Fitting and Minimization

- ◆ Development of Minuit C++ almost complete (thanks to Matthias)
- ◆ Working also on a fitting and minimization framework
 - Based on Minuit but with possibility to change Minimizer engine at run time
 - Fitting interface for Chi2 and Likelihood fits
 - Should be available in one of the next version of SEAL
 - Should contain also Python wrapper for interactive usage (based on LCG C++ dictionary)

Conclusions

- ◆ Next meeting

- In 3 weeks time ?

- » Possible date Tuesday the 18th May 2004 ?

- ◆ AOB