SEAL-ROOT
Math Plans for 2005

Math work package

Andras Zsenei, Anna Kreshuk, Lorenzo Moneta, Eddy Offermann

LCG Application Area Internal Review, 30 March, 2005

Math Work Package

- Main responsabilities for this work package:
  - Basic Mathematical functions
    - *TMath*, SEAL *MathCore*
  - Functions and Fitting
    - Parametric function classes (*TF1*)
    - Minimizers (*Minuit, Fumili*) and linear and robust fitters, quadratic programming, etc..
  - Random Numbers
  - Linear Algebra
  - Physics Vector
- Not discussed now, but still relevant :
  - Histograms
  - Statistics (confidence level)
  - Neural Net, multivariate analysis, etc..

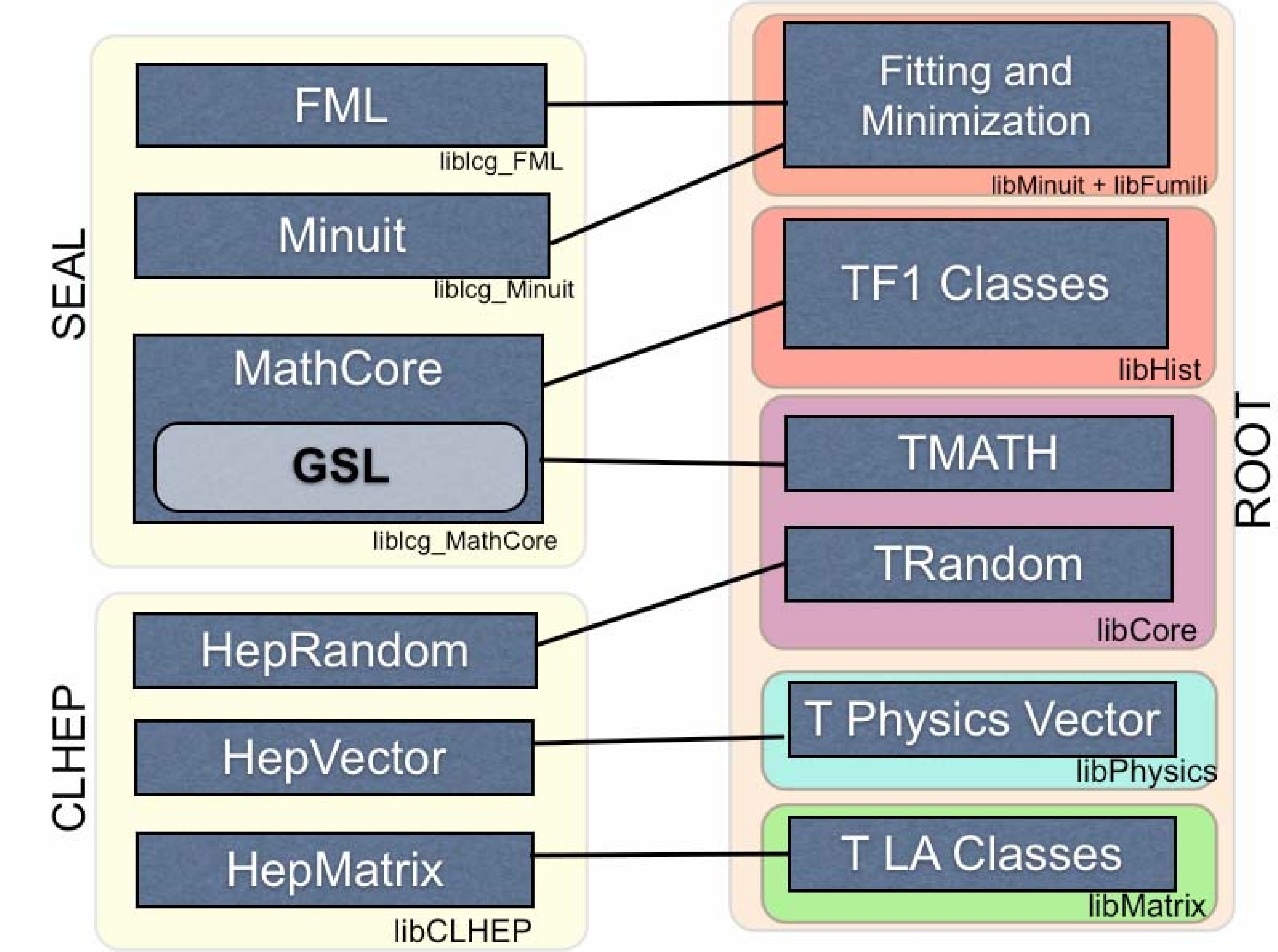
Outline

- Preliminary proposal for SEAL - ROOT integration and evolution for short-medium term
  - SEAL *MathCore* vs *TMath*
  - Improvements for Function classes (*TF1*)
    - integration of SEAL *MathCore* numerical algorithms
  - Fitting and Minimization
    - integration of SEAL Minuit and SEAL Fitting framework
- Possible proposal for merging ROOT and CLHEP
  - Random numbers
  - Physics Vectors
  - Linear Algebra
- Milestones
- Conclusions

SEAL Math Lib Contents

- ***MathCore***
  - library with the basic Math functionality
  - used GSL in implementation
  - interfaces could be re-implemented using another library
  - design reviewed by CMS
- ***Minuit***
  - re-implementation of Minuit in C++
  - stand-alone package (no ext. dependencies)
- ***FML* (Fitting and Minimization Library)**
  - defines some generic interfaces for fitting and minimization
  - use Minuit

SEAL, CLHEP and ROOT Math Libraries



Differences *TMath-MathCore*

- *TMath* and *MathCore* both contain large variety of special and statistical functions
  - *MathCore* implements proposed interfaces to C++ standard for special functions
  - *MathCore* has a more complete set (~ 70 functions)
- *MathCore* uses GSL in implementation
  - good library but with GPL license
- *TMath* has its own implementation
  - some functions are coming from Numerical Recipes
    - also license problem and not quality as GSL
- Need to develop our implementation or take one from an open source free license
  - some available for basic functions (i.e. *cephes*)

Proposal for Math Functions

- Separate functions according to use
  - ROOT *mathcore* CVS directory with most used functions:
    - Beta and Gamma functions, Erf, etc..
      - from these we can derive majority of the functions used in statistics (i.e. Chi2 probability)
    - we keep and maintain these implementations
    - part of ROOT *libCore* (size should be ~ 500 kB)
  - A new math directory with the other less used functions:
    - Bessel, Legendre polynomial, Hypergeometric, etc..
      - use GSL for the implemention as in SEAL *MathCore*
    - build as an independent *libMath*
    - distribute with GPL license

Special Functions

- Use new SEAL interfaces (C++ standard proposal)
- Have a separate namespace (*specfunc*)
  - easy transition in the future when they will be in std namespace
  - Example of new interfaces and how to keep backward compatibility

```
namespace specfunc {
   double cyl_bessel_i (double nu, double x);
   ......
}
namespace TMath {
   double BesselI(double x, int n) {
      return cyl_bessel_i(static_cast<double>(n),x);
   }
   .....
}
```

Further TMATH Improvements

- Possible work for long term (end of the year)
  - use STL for sorting algorithm, min/max etc.
    - evaluate the performances and eventually drop the old algorithms
  - use *std::vector* in interface in addition to C arrays
    - performances are the same when accessing elements
  - separate also statistic functions acting on containers:
    - mean, RMS, median, skewness
    - have template functions for all of them
    - move in a new Statistics library (*libStat*)

Function Classes and Algorithms

- SEAL *MathCore*
  - Generic function interfaces (i.e *IParametricFunction*)
  - Classes for numerical algorithms
    - Integration, Derivation, Root Finders, Interpolation
  - separation between functions and algorithms
- ROOT Function classes
  - parametric function classes (i.e *TF1*)
    - mathematical functionality but also Fitting, Random and plotting functionality
    - algorithms (integration, derivation) implemented inside the *TF1* and derived classes

Proposal for Functions

- Separate plotting functionality from TF1
  - make independent of plotting (use *TVirtualHistPainter*)
- Finalize design of function interfaces started in SEAL
  - Implement *TF1* using new interfaces of *MathCore*
- Separate implementation of the numerical algorithms from the function classes
  - have integration, derivation in separate classes
    - have different implementations for the same algorithm
      - develop and maintain the basic implementation
      - have also alternatives based on GSL (or others libs)
  - put basic implementations in *libCore* while others in the extended new Math library (same as TMath)
    - TF1 classes will use the interfaces and will not have direct dependency on *libMath*

Fitting and Minimization

- Fitting in ROOT goes through the *TVirtualFitter*
  - abstract class but designed for Minuit
    - mixes fitting and minimization
  - *TVirtualFitter* implementations:
    - *TFitter* (*TMinuit*), *TLinearFitter* and *TFumili*
      - code duplication (for example in setting parameters)
- In SEAL we have FML (fitting framework)
  - FML defines generic interfaces for fitting and minimization
    - one implementation is new C++ Minuit (T*MinuitCpp*)
  - We should start from SEAL fitting libraries and aim to have a real fitting framework in ROOT
  - combine with proposed re-design of the Function classes

Fitting Proposal

- **Short term (first development release of ROOT 5)**
  - **have a new Fitter class (*TMinuitCpp*) implementing the *TVirtualFitter* interface and using new C++ Minuit**
  - **some work already done by Matthias Winkler**
  - **continue evaluation and plan to make default engine**
- **Medium/Long term (end of the year):**
  - **integrate FML in ROOT, redesigning and adapting to the new Function interfaces of *MathCore***
  - **make best solution to integrate all existing implementations:**
    - **Linear and Robust Fitters, Fumili, Minuit, quadratic optimizer and also *TMultiDimFit* and *TFractionFitter***
    - **isolate common functionality in a *libFitter* library**
    - **various plug-in libraries for the various implementations**

CLHEP

- CLHEP packages (which are the most used ones):
  - Random
  - Vector (Physics Vectors)
  - Matrix
- Similar functionality exist in ROOT
  - *TRandom* classes
  - *TVector2 (3)* and *TLorentzVector* classes
  - new ROOT linear algebra package
- Major difference is *TObject* inheritance for ROOT classes
- Otherwise there is lots of duplication
  - problem for long term maintenance
  - make sense aiming to merge in a new library

Random (CLHEP vs ROOT)

- CLHEP Random package
  - Nice separation engine - distributions
    - abstract class for engines with various implementations
      - *Ranmar, Ranecu, Mersenne-Twister, RanLux,...*
    - classes for each distribution
      - *RandFlat, RandExponential, RandGauss, etc*
- ROOT Random:
  - base class (*TRandom*) with default engine
    - *rndm()* from Cernlib
      - fast generator but with small period (10**8) and obsolete in Cernlib
    - base class defines functionality for random distributions
    - possibility to store in a file (*TRandom.Write()*)
  - *TRandom2* (based on *rdm2()*)
  - *TRandom3* (based on Mersenne-Twister generator)
    - both inheriting from *TRandom*

Proposal for Random

- Improve ROOT random (for first dev. release)
  - make *TRandom3* the default engine and rename it (*TMersenneTwister*)
  - add missing engines from CLHEP
    - *RanLux, HepJames* (RANMAR)
  - add more distributions (taking from CLHEP and/or GSL):
    - Gamma, Chi2, LogNormal, F-dist, t-dist, geometric, etc..
  - have still Random as part of the core Math packages (in ROOT *libCore*)
- Proposed work for the long term:
  - Evaluate random number proposal to C++ standard
    - template classes on Engines and Distribution Type
    - a similar implementation already exists in Boost
  - re-implement using the new proposed interface?

CLHEP & ROOT Physics Vectors

- CLHEP
  - *Hep2Vector, Hep3Vector, HepLorentzVector*
  - *HepRotation, HepLorentzRotation, HepBoost*
- ROOT
  - *TVector2, TVector3, TLorentzVector*
  - *TRotation, TLorentzRotation*
  - originated from CLHEP in `98 and developed in parallel
- Large overlap of functionality (> 90 %)
- Bloat interface for CLHEP (from CLHEP-ZOOM merger)
  - lots of duplications (*getX()* and *x()*)
- Slightly more functionality in CLHEP classes
  - nearness concept, ordering, etc..
    - probably reside better in some utility classes

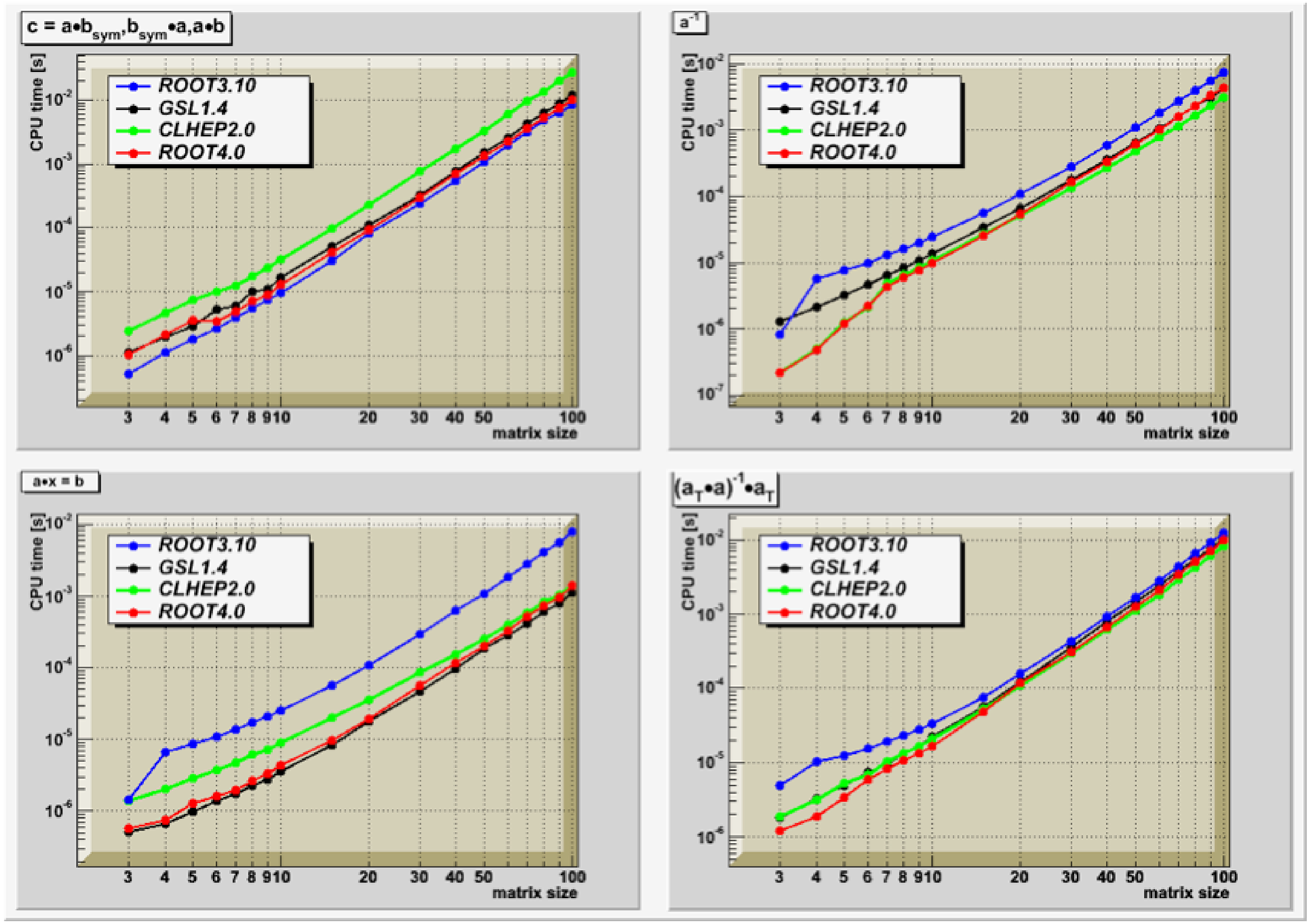
Physics Vector Proposal

- Short term work proposal
  - base on existing ROOT classes
  - eventually add missing functionality from CLHEP
- Medium/Long term (end of the year) work proposal
  - improve interface of physics vector classes merging CLHEP and ROOT
    - good occasion for a clean-up and redesign
      - CLHEP interfaces were designed more than 10 years ago
    - evaluate pro/cons of *TObject* inheritance
    - need to involve some of CLHEP authors and experiments
      - it would require some time
    - we must consider that these classes are heavily used
      - problem of migration to new classes

Linear Algebra

- More functionality in ROOT Linear Algebra
  - decompositions for solving LA systems
  - support for sparse matrices
  - support for external data storage
- Both have optimized support for small matrices
  - pre-allocation on the stack up to 6x6 matrix and optimized inversion
- Proposal is to base on ROOT Linear Algebra
- Consider to move in the long term to template classes for supporting complex matrices
- Continue detailed evaluation with other LA packages
  - follow evolution of new GLAS (Boost) project

Linear Algebra Tests

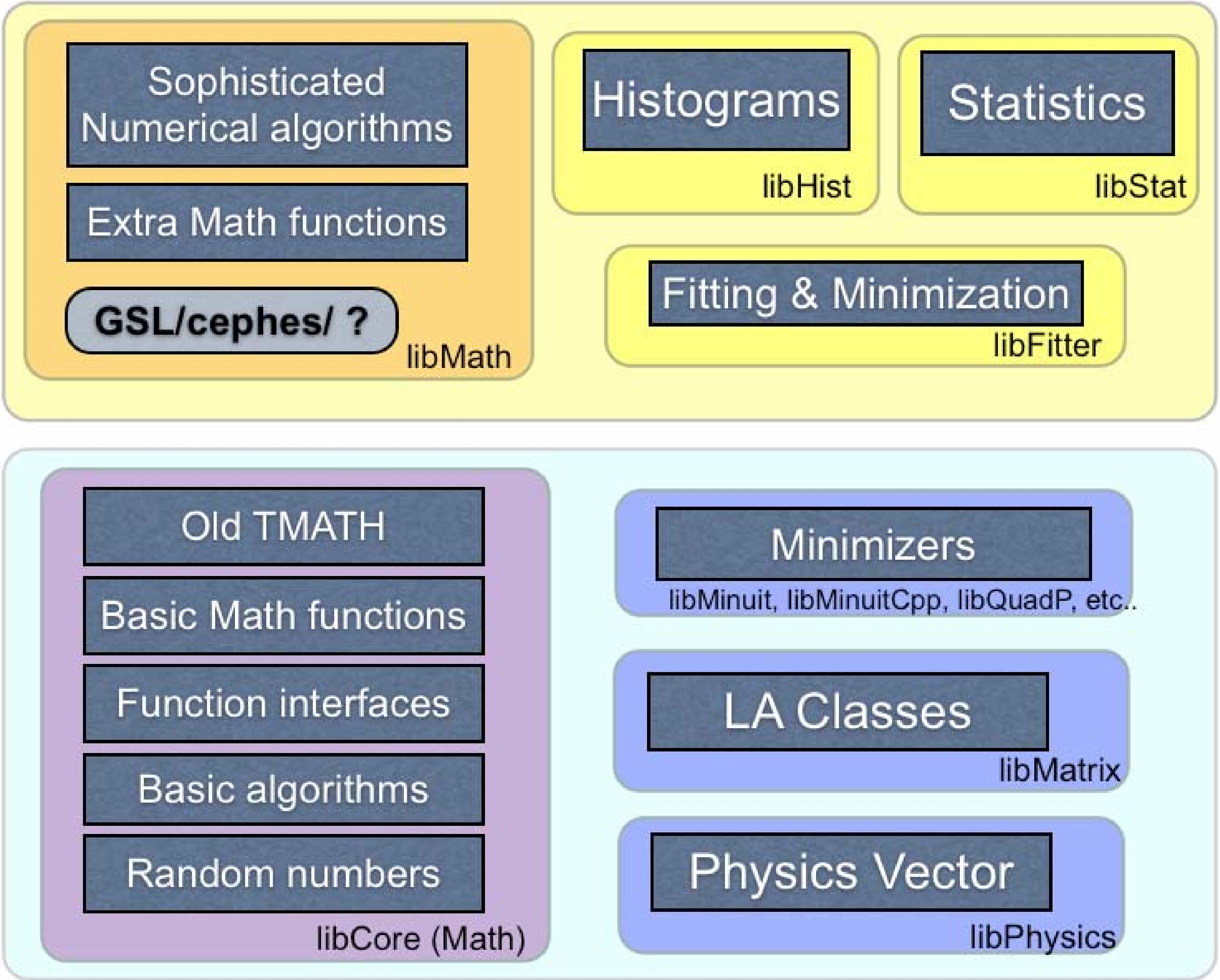


- **Important to continue detailed performance evaluation of existing LA packages**
  - **test also in real experiment applications**

Additional Activities

- Combine in a Statistic library *(libStat*)
  - statistical functions and classes to estimate confidence level (*TLimit*, *TFeldmanCousin, TRolke)*
- Improve multivariate analysis methods
  - neural net classes (*TMultiLayerPerceptron*) and add new algorithms (*i.e.* Boost Decision Trees)
  - *TPrincipal*, cluster finders algorithms
- Improve Histogram library
  - add peak finders in 2D
- Possible new Math functionality to be added:
  - quasi-random number generators
  - Monte Carlo integration methods and tools (*Foam, PI*)
    - ongoing discussions with S. Jadach and F. Krauss

Proposed new Math Structure



Sophisticated Numerical algorithms

Extra Math functions

GSL/cephes/ ?

libMath

Histograms

libHist

Statistics

libStat

Fitting & Minimization

libFitter

Old TMATH

Basic Math functions

Function interfaces

Basic algorithms

Random numbers

libCore (Math)

Minimizers

libMinuit, libMinuitCpp, libQuadP, etc..

LA Classes

libMatrix

Physics Vector

libPhysics

Math Milestones

- **30 June 2005**
  - new *mathcore* CVS directory in ROOT with the basic math functions
  - new *libMath* containing GSL wrappers (from SEAL)
  - Integration of SEAL Minuit in ROOT (*TMinuitCpp*)
  - Improve ROOT random number package
- **30 Sep 2005 (ROOT Workshop)**
  - Complete re-factor of *TF1* classes
  - Design of new Physics Vector interfaces
  - Design of new Fitting framework
- **15 Dec 2005**
  - first release of new Physics Vector library
  - first release of new Fitter library

Math Conclusions

- Smooth integration of SEAL packages in ROOT
  - expect to finish migration for end of the year
  - opportunity to re-factor and redesign functions and fitting in ROOT
- Merger also with CLHEP
  - lots of duplicated functionality ROOT-CLHEP
  - make sense for long-term maintenance to have a single package used by the experiments
- Propose to build new physics vectors library
  - occasion for redesigning and merging the functionality
- Important to develop in collaboration with the experiments
  - need agreement in using the new libraries