Developments of Mathematical Software Libraries for the LHC experiments

Lorenzo Moneta

CERN/ PH-SFT

On behalf of LCG SEAL/MathLib team

Outline

- Introduction to the MathLib project
- Requirements for mathematical libraries
- Design for C++ MathLib's
- Tests and validation of existing libraries
- Libraries for fitting and minimization (MINUIT)
- Summary and Conclusions

The MathLib Project

- Project originated from LCG RTAG on Mathematical libraries
- It is part now of the LCG SEAL project
- Project goals are:
  - provide coherent set of Mathematical Libraries to end-users and developers of LHC experiments
  - avoid maintenance burden of similar libraries
- Requirement to use the same libraries in all environments
  - simulation, reconstruction and analysis
  - from C++ and interactively (Python, CINT)
- Development done in collaboration with the LHC experiments and other LCG projects (ROOT)

General Requirements for Math Lib's

- Set of components with as little coupling as possible
- Allow dependency on C++ Standard Library
  - use std::vector and std::complex
- Allow dependency on external products if
  - they provide directly needed functionality
  - meet support and quality standard specified by LCG
- Refrain from duplicating functionality already present in STL
  - vector operations, searching and sorting algorithms, etc..)
- Avoid non - mathematical functionality

Math Libraries Contents

- What is the the needed functionality ?
- A good starting point is what CERNLIB offers
  - but skipping what exists already in STL
  - skip also HEP kinematics and simulation
- Produced an inventory of functions and algorithms
  - group them by related functionality
    - » follow GSL organization
  - 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

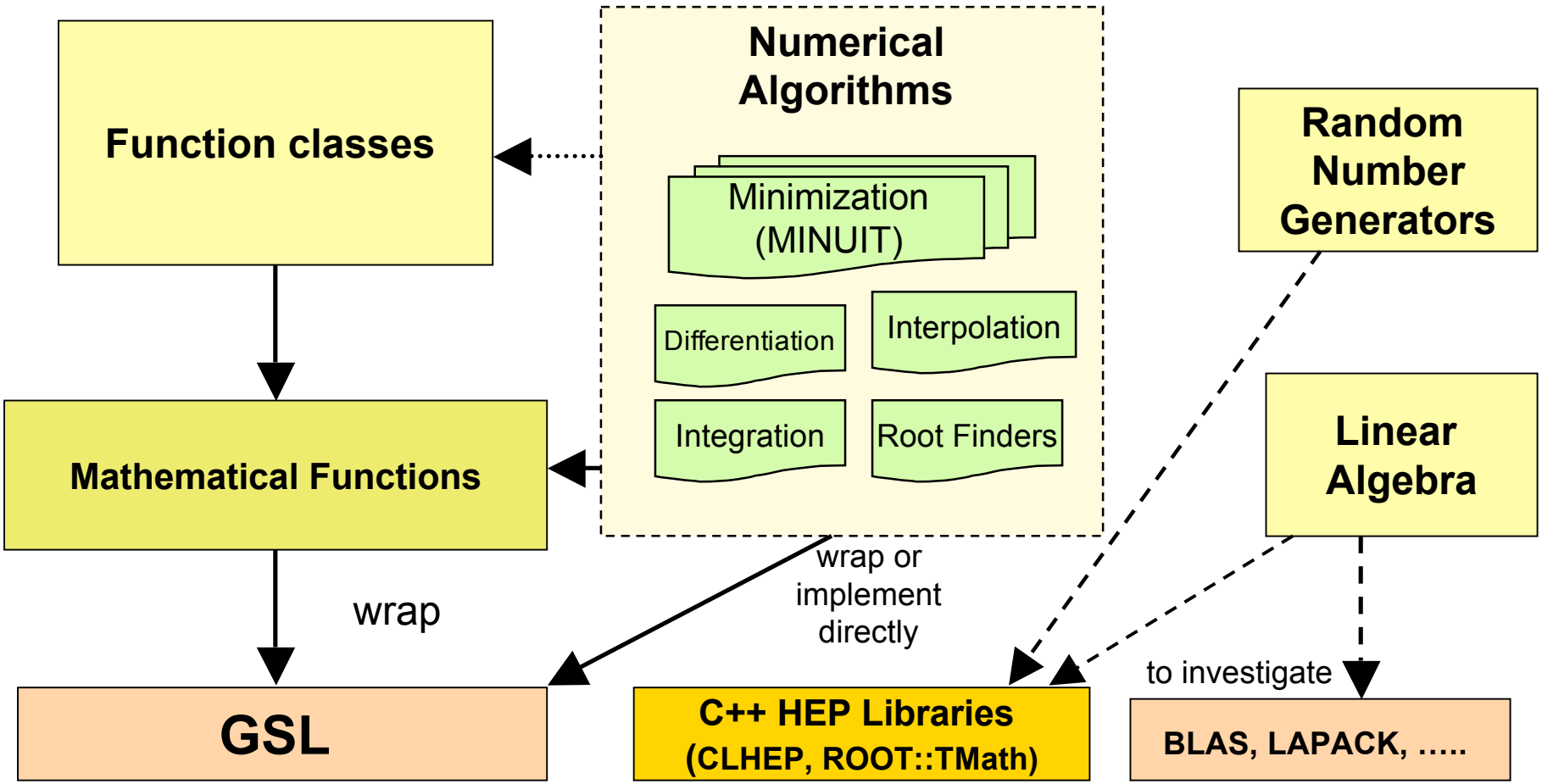
---

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
- 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
- Algorithms
  - Numerical Integration and Differentiation, Function Minimization, Root Finders, Interpolators, etc…
- Random number generators

C++ MathLib


Function classes
Numerical Algorithms
Minimization (MINUIT)
Differentiation
Interpolation
Integration
Root Finders
Random Number Generators
Linear Algebra
Mathematical Functions
wrap
wrap or implement directly
to investigate
GSL
C++ HEP Libraries (CLHEP, ROOT::TMath)
BLAS, LAPACK, ...

Mathematical Functions

- 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 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 a very small overhead

Example of Free Functions

- Special functions (Bessel)

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

- Implementation using GSL

```
#include “gsl/gsl_sf.h”

mathlib::bessel_I ( double l, double x) {
        return gsl_sf_bessel_In (l, x);
}
```

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
- Need for function operations
  - arithmetic operations, composition, convolution
- 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

Algorithms

- Algorithms will use abstract function interfaces
  - No direct dependency on the function library
- 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..
  - this would avoid virtual function call
- 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
  - have also a method directly passing C function pointers to avoid adapters

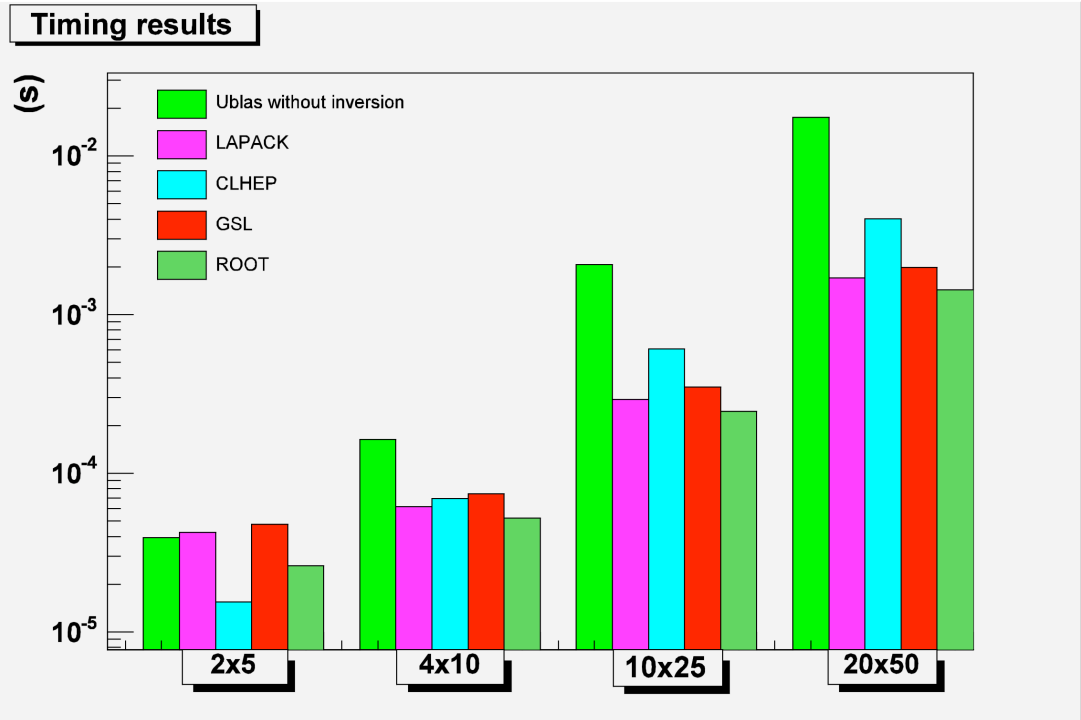
```
class Integrator {
…..
// generic integration method
template < class Function >
double integrate ( const Function & f, double a, double b);

// specialization for IFunction
double integrate ( const IFunction & f, double a, double b);

// method using function pointer (same signature as GSL function)
typedef   ( * CF )   ( double , void * );
double integrate ( CF f,  double * params, double a, double b);
……
};
```

Linear Algebra

- Library with matrix and vector classes
  - use C++ operator overloading to implement vector/matrix operations
- 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 use it for Linear Algebra studies (see later)
  - Have a version based on a customized implementation in MINUIT

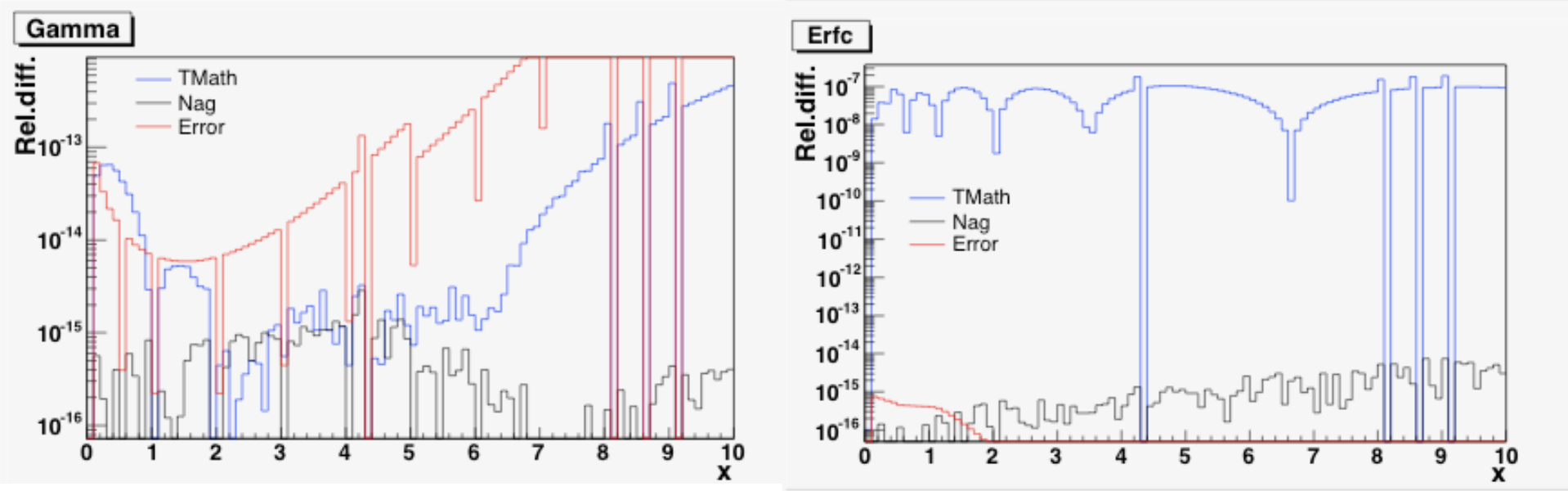
Linear Algebra studies

- 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)



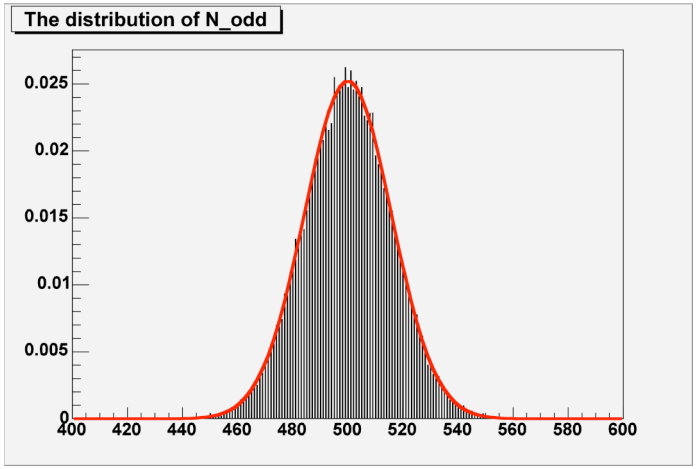
Evaluation of existing libraries (GSL)

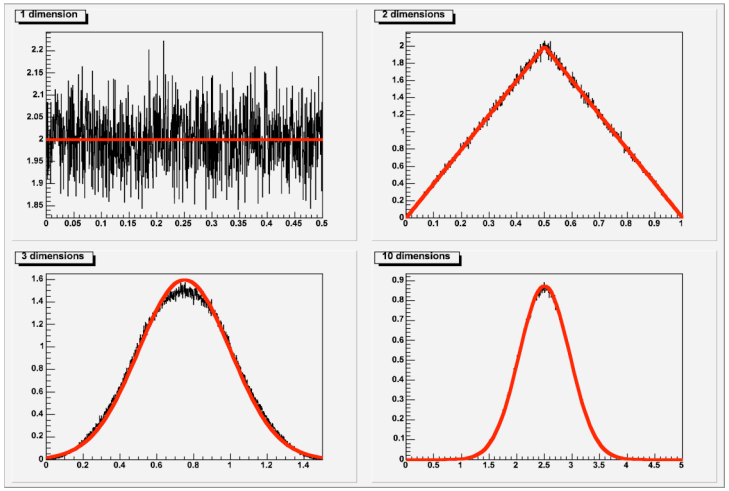
- Test numerical accuracy and time performances of GSL, NAG and ROOT
  - Compare special functions (Bessel, Gamma, Erf) and some statistical functions (e.g. Chi2 probability)
- Good numerical results obtained from GSL



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





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

Summary

- Started providing support in Math Libs for LHC experiments
  - Inventory of functions and algorithms
- Produced a design for C++ mathematical libraries
- Will start with an implementation based mainly on GSL
- Validation tests of GSL have not shown so far major defects
- Delivered C++ MINUIT with same functionality as in Fortran
- Continue and complete the developments of MINUIT
  - Implement Fumili, specialized minimizer for least-square and likelihood fits
  - Adapt to be easily integrated into new MathLib C++
- Starting providing libraries to experiments
  - 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

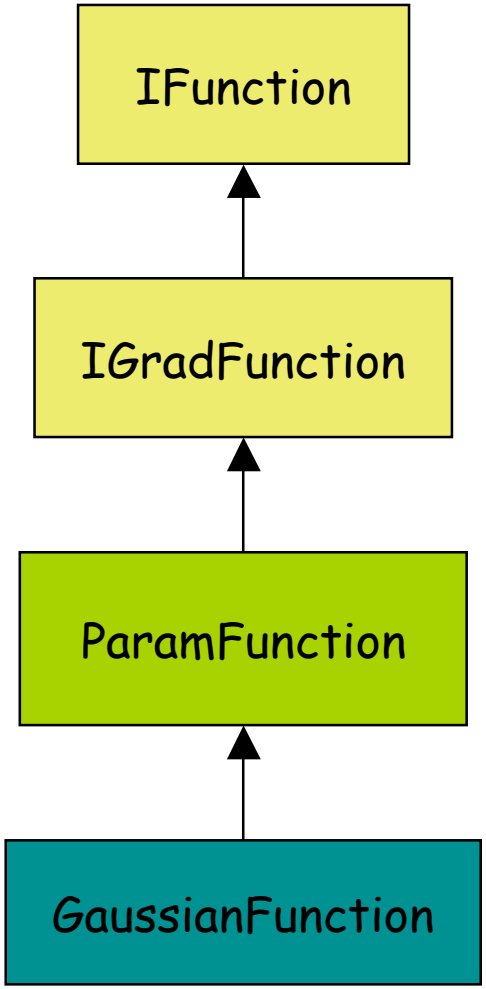
    with documentation (User Guide and minimization tutorial)

    and links to download code (can be built easily with configure/make)
- Mailing lists:
  - forum-mathlib@cern.ch
  - forum-minuit@cern.ch
- Acknowledgments:
  - SEAL team members
  - W. Brown, R. Brun, M. Paterno

Back-up Slides

C++ Function Design

- Abstract Function interface
  - Defines only : double operator() (double x)
  - Have a 1D and a multidimensional interface
    - » double operator() (const std::vector<double> & x)
- Abstract Gradient function
  - Defines in addition: double gradient (double x)
- Base Parametric function class
  - Has parameters as member
    - » Defines set/getParameters()
  - Not abstract, so derived classes do not need to re-implement all methods
  - Have classes implementing gradient and parameter gradient
- Sets of concrete classes implementing pre-defined functions:
  - Gaussian, Exponential, etc…



Evaluation of existing libraries (2)

- Timing performance evaluating special functions