Latest developments in ROOT::Math

Magdalena Sławińska

12th October 2007
Outline

1. Introduction

2. Numerical Integration
   - New classes for multidimensional integration
   - Comparison of performance
   - Interface

3. Implementing special functions
   - Base functions
   - Probability distributions

4. On-line documentation
   - Maths @ TWiki
   - On-line inventory
Restructuring and Extending Mathematical Libraries

1. Revise already implemented functions and algorithms:
   - Testing accuracy, time performance, domains,
   - Crosschecking with well-established implementations: GSL, Mathematica
   - (Re)implementing existing ones.

2. Add useful routines.

3. Provide convenient user interfaces.

4. Reduce dependency on GSL libraries and other ROOT classes (TF1).
What I did

1. New classes for multidimensional integration, using:
   - Adaptive cubature
   - Monte Carlo integration

2. Reimplementation of major special functions and probability distributions

3. On-line documentation (TWiki, upgrade of Inventory), tutorials.
Adaptive cubature

- Checking existing algorithms (ROOT::TF1, TFoam, HintLib)
  - → the best applies Genz-Malik cubature
  - → the best implementation in TF1 class
- Extend functionality of TF1::IntegralMultiple.
  - independence on TF classes and hist library,
  - usage for free functions,
  - and arbitrary number of dimensions (not only 2 or 3).
  → IntegralMultiDim class
IntegralMultiDim class - usage

```cpp
ROOT::Math::WrappedParamFunction<> fptr1(&Func, NDim);

ROOT::Math::IntegratorMultiDim ig1;

ig1.SetFunction(fptr1);

ig1.Integral(xmin[], xmax[]);
```
Applying Monte Carlo algorithms

- C++ interface to algorithms from GSL
- MC algorithms for the first time in ROOT (these from RooFit are not universal)
- Good functionality (not restricted to TF objects)

→ GSLMCIintegrator class
The usage of GSLMCIIntegrator class

\begin{verbatim}
ROOT::Math::WrappedParamFunction<> fptr(&Function, NDim);
ROOT::Math::GSLMCIIntegrator ig1();
\end{verbatim}

Setting algorithm

\begin{verbatim}
ig1.SetType(ROOT::Math::MCIntegration::VEGAS);
ig1.SetMode(ROOT::Math::MCIntegration::IMPORTANCE);
\end{verbatim}

Setting additional parameters

\begin{verbatim}
VegasParameters param;
param.iterations = 2;
ig1.SetParameters(param);
\end{verbatim}

\begin{verbatim}
ig1.SetFunction(fptr);
ing1.Integral(xmin[], xmax[]);
\end{verbatim}
Adaptive Cubature vs Monte Carlo

**Figure:** Achieved accuracy: 0.01
Using the **plug-in manager** one can load GSL implementation in MathMore
Special functions

Major special functions

- commonly used in physical calculations,
  - (upper) incomplete gamma,
  - incomplete beta
- used for implementing other functions:
  - $\chi^2$ (used in statistical tests performed many times)
  - $F$-distribution, Student-$t$ distribution, etc (for statistical analysis).

After testing several implementations decided on CEPHES

- free usage netlib
- GSL need license
- NR copyrighted “no transfer or distribution”
- many other pros...
Gamma and related functions

Old TMath

- Gamma, LnGamma

New MathCore

- Implementation from NR
- Independent on GSL
Gamma and related functions

Old TMath

- **Gamma, LnGamma**

- Implementation from NR
- Domains: \((0, \infty)\)

New MathCore

- Independent on GSL
- Domains extended to \((-\infty, \infty) \setminus \mathbb{Z}_-\)
Gamma and related functions

Old TMath

- **Gamma, LnGamma**

  - Implementation from NR
  - Domains: \((0, \infty)\)
  - Efficiency \(11 \cdot 10^{-5}\) (s/100 calls)

New MathCore

- Independent on GSL
- Domains extended to \((-\infty, \infty) \setminus \mathbb{Z}_-\)
- Efficiency \(5 \cdot 10^{-5}\) (s/100 calls)
Gamma and related functions

Old TMath

New MathCore

- Upper incomplete gamma

- Implementation from NR

- Implementation independent of GSL
Gamma and related functions

Old TMath                      New MathCore

- Upper incomplete gamma

- Implementation from NR
- Efficiency $11 \cdot 10^{-5}$ (s/100 calls)

- Implementation independent of GSL
- Efficiency $7 \cdot 10^{-5}$ (s/100 calls)
Beta and related functions

Old TMath

- **Beta function**

New MathCore

- Beta already implemented in TMath
- Implementation based on CEPHES
Beta and related functions

Old TMath

- Beta function
- Beta already implemented in TMath

New MathCore

- Implementation based on CEPHES
Beta and related functions

Old TMath

- Incomplete Beta
- Implementation in TMath from NR

New MathCore

- Implementation independent of GSL
Beta and related functions

Old TMath

New MathCore

- Incomplete Beta

- Implementation in TMath from NR
  - Efficiency $4 \cdot 10^{-4} \text{(s/100 calls)}$

- Implementation independent of GSL
  - Efficiency $4 \cdot 10^{-4} \text{(s/100 calls)}$
Beta and related functions

Old TMath

- Incomplete Beta
  - Definition

New MathCore

- Implementation in TMath from NR
  - Efficiency $4 \cdot 10^{-4} \text{ (s/100 calls)}$

- Implementation independent of GSL
  - Efficiency $4 \cdot 10^{-4} \text{ (s/100 calls)}$
Error function

Old TMath
- implementation from NR

New MathCore
- from CEPHES library
Error function

Old TMath

- implementation from NR

New MathCore

- from CEPHES library

- Efficiency around 0 better in CEPHES (10%), tails better in TMath (factor 2)
**Error function**

### Old TMath
- Implementation from NR

### New MathCore
- From CEPHES library

- Efficiency around 0 better in CEPHES (10%), tails better in TMath (factor 2)
- Improvement in accuracy! (single $\rightarrow$ double precision)

▶ erfc comparison
Probability distributions & cumulatives

Using new implementation of `inc_gamma` and `inc_beta` one can reimplement

- chi^2 comparison
- F comparison
- Student-t comparison

independently of GSL and NR!

The new implementation in MathCore uses

- base functions from CEPHES library,
- reflexive property for `inc_beta` function to calculate complementary cumulatives
Description & Documentation

An online reconnaissance over Mathematical Libraries:
http://root.cern.ch/twiki/bin/view/ROOT/MathematicalLibraries

Information about existing algorithms and functions:
http://root.cern.ch/twiki/bin/view/ROOT/MathTable

Reference to be found at:
Mathematical Libraries in ROOT

The aim of Math libraries in ROOT is to provide and to support a coherent set of mathematical and statistical functions. The latest developments have been concentrated in providing first versions of the MathCore and MathMore libraries, included in ROOT v5.08. Other recent developments include the new version of MINUIT, which has been re-designed and re-implemented in the C++ language. It is integrated in ROOT. In addition, an optimized package for describing small matrices and vector with fixed sizes and their operation has been developed (SMatrix).

The structure is shown in the following picture.
Updated Inventory

Inventory of Mathematical Functions and Algorithms

<table>
<thead>
<tr>
<th>Functions and Polynomials</th>
<th>Numerical Methods</th>
<th>Random Numbers and Distributions</th>
<th>Others</th>
</tr>
</thead>
<tbody>
<tr>
<td>Special Functions</td>
<td>Integration</td>
<td>Random Number Generator</td>
<td>Linear Algebra</td>
</tr>
<tr>
<td>Polynomials</td>
<td>Differentiation</td>
<td>Random Number Distribution</td>
<td>Differential Equations</td>
</tr>
<tr>
<td>Function Approximations</td>
<td>Minimization</td>
<td></td>
<td>FFT</td>
</tr>
<tr>
<td></td>
<td>Root-Finding</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Interpolation</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Special Functions

Routines for evaluating Special functions

Bessel Functions of various types

Regular cylindrical functions

Bessel functions of various orders

ROOT, GSL, Cernlib

Irregular cylindrical functions

Bessel functions of various orders

ROOT, GSL, Cernlib

Regular modified cylindrical

Bessel functions of various orders

ROOT, GSL, Cernlib

Irregular modified cylindrical

Bessel functions of various orders

ROOT, GSL, Cernlib

Regular spherical functions

Bessel functions of various orders

ROOT, GSL, Cernlib

Irregular spherical functions

Bessel functions of various orders

ROOT, GSL, Cernlib

Examples and tutorials

Clausen function

Clausen integral function

GSL, Cernlib
The work is done...

Thank You!
gamma function

\[ \Gamma(z) \equiv \int_0^\infty dt \ t^{z-1} e^{-t}, \quad a > 0 \]
(upper) incomplete gamma function

\[
P(a, z) \equiv \frac{1}{\Gamma(a)} \int_0^\infty dt \ t^{a-1} e^{-t}, \quad a > 0
\]
beta function

\[ B(z, w) \equiv \int_0^1 dt \ t^{z-1}(1 - t)^{w-1} \]

\[ B(z, w) = \frac{\Gamma(z)\Gamma(w)}{\Gamma(z + w)} \]
incomplete beta function

\[
\text{inc\_beta}(x; a, b) \equiv \frac{1}{B(a, b)} \int_0^x dt \ t^{a-1}(1 - t^{b-1}) \quad a, b > 0
\]

\[
1 - l(x; a, b) = l(1 - x; b, a)
\]
**Figure:** Comparison with Mathematica Gamma[x]
Figure: Comparison with Mathematica GammaRegularized[a, x]
Figure: Comparison with Mathematica Beta[a, x]
Figure: Comparison with Mathematica BetaRegularized[x,a,b]
Figure: Comparison with Mathematica Erfc[x]
Figure: Comparison with Mathematica ChiSquareDistribution[x]
Figure: Comparison with Mathematica FRatioDistribution[x]
Figure: Comparison with Mathematica