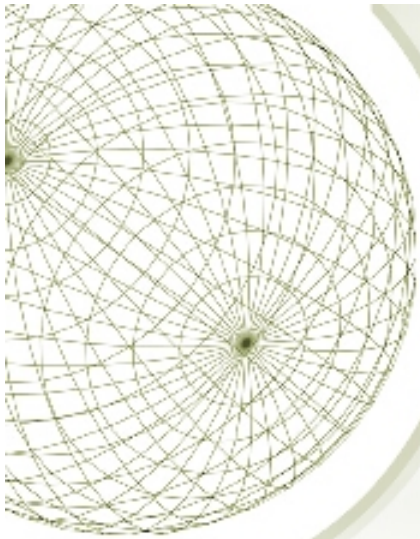
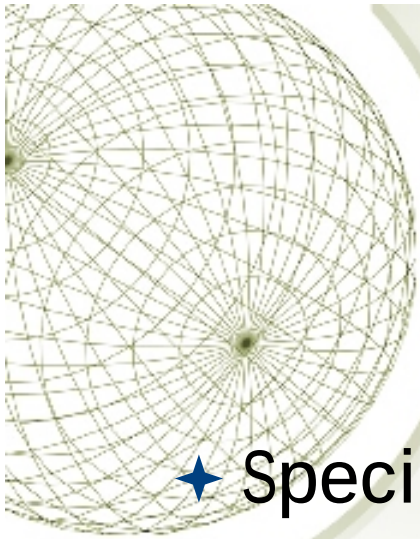


MathMore

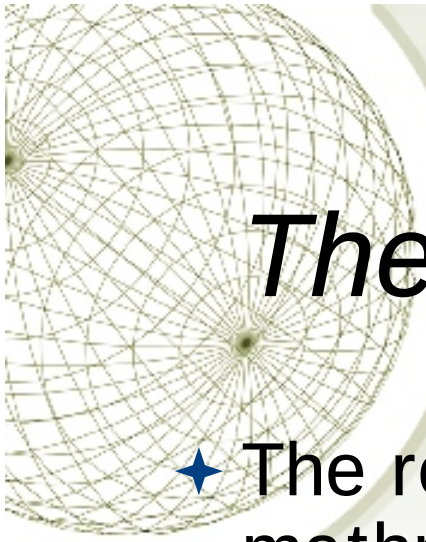
Lorenzo Moneta, Andràs Zsenei

ROOT Workshop 30/9/2005



MathMore components

- Special functions
- Statistical functions
- Generic interfaces
- Derivation
- Integration
- Interpolation
- Root finding
- Chebyshev polynomials



The Current Implementation

- The relevant GSL code extracted into mathmore/src/gsl-xxx and compiled automatically
  - A GSL tar file is in CVS with the extracted code
  - Works on all supported platforms (thanks to Bertrand Bellenot for the Windows port)
- Easily maintainable and updateable compared to direct copy of the algorithms into ROOT classes

Special Functions

- Defined in the N1687 Technical Report on Standard Library Extensions
  - http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2004/n1687.pdf
- Basic functions in mathcore:
  - Error functions, gamma functions
- The less used and those that use GSL implementation are in mathmore
  - Bessel functions, elliptic integrals, Legendre polynomials etc
- Possibility to migrate functions between muthcore and mathmore transparently for the user

Special Functions cont'd

- Free functions following the C++ standard's naming convention (N1687)
- Trivial use:

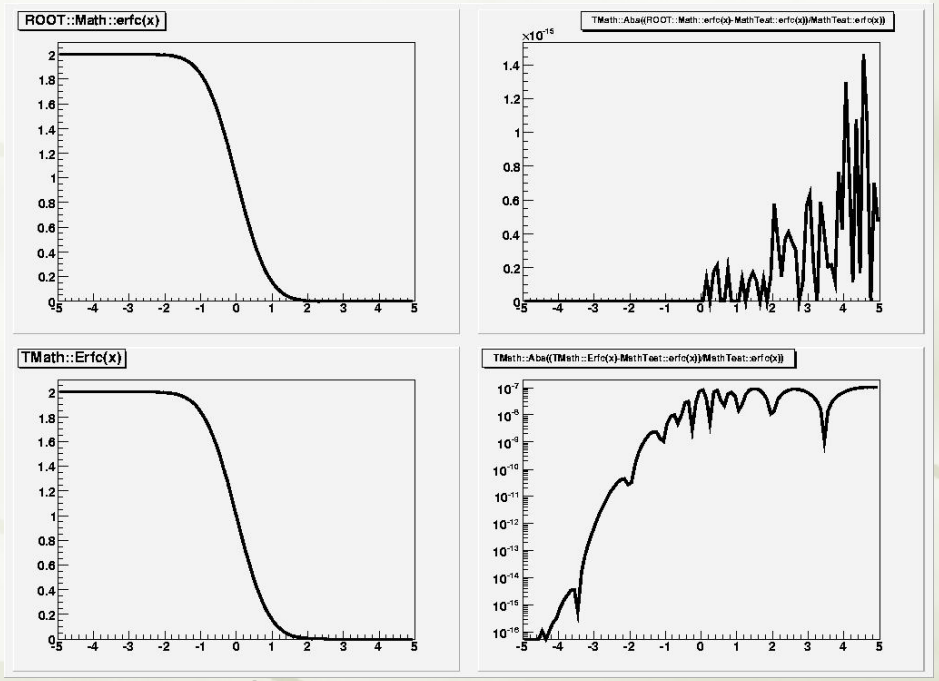
```
root [0] gSystem->Load("libMathMore.so");
root [1] ROOT::Math::cyl_bessel_i(1.2, 2.4)
(double)2.05567401212170076e+00
```

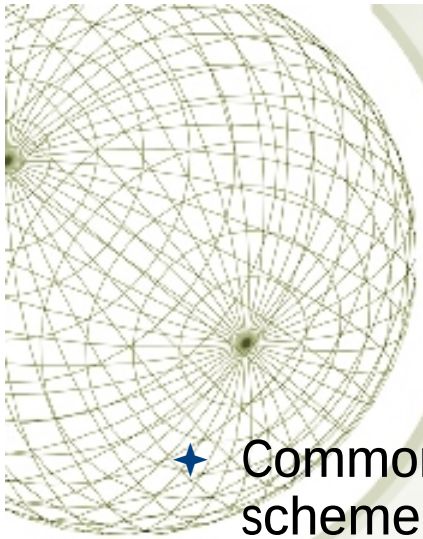
Mathematical Functions tests

- New functions improve on the precision of TMath
- Extensive tests of numerical accuracy, comparisons with other libraries (Mathematica, Nag)

ROOT::Math::erfc and relative difference compared to Mathematica ($\Delta \approx 10^{-15}$)

TMath::Erfc and relative difference compared to Mathematica ($\Delta \approx 10^{-7}$)



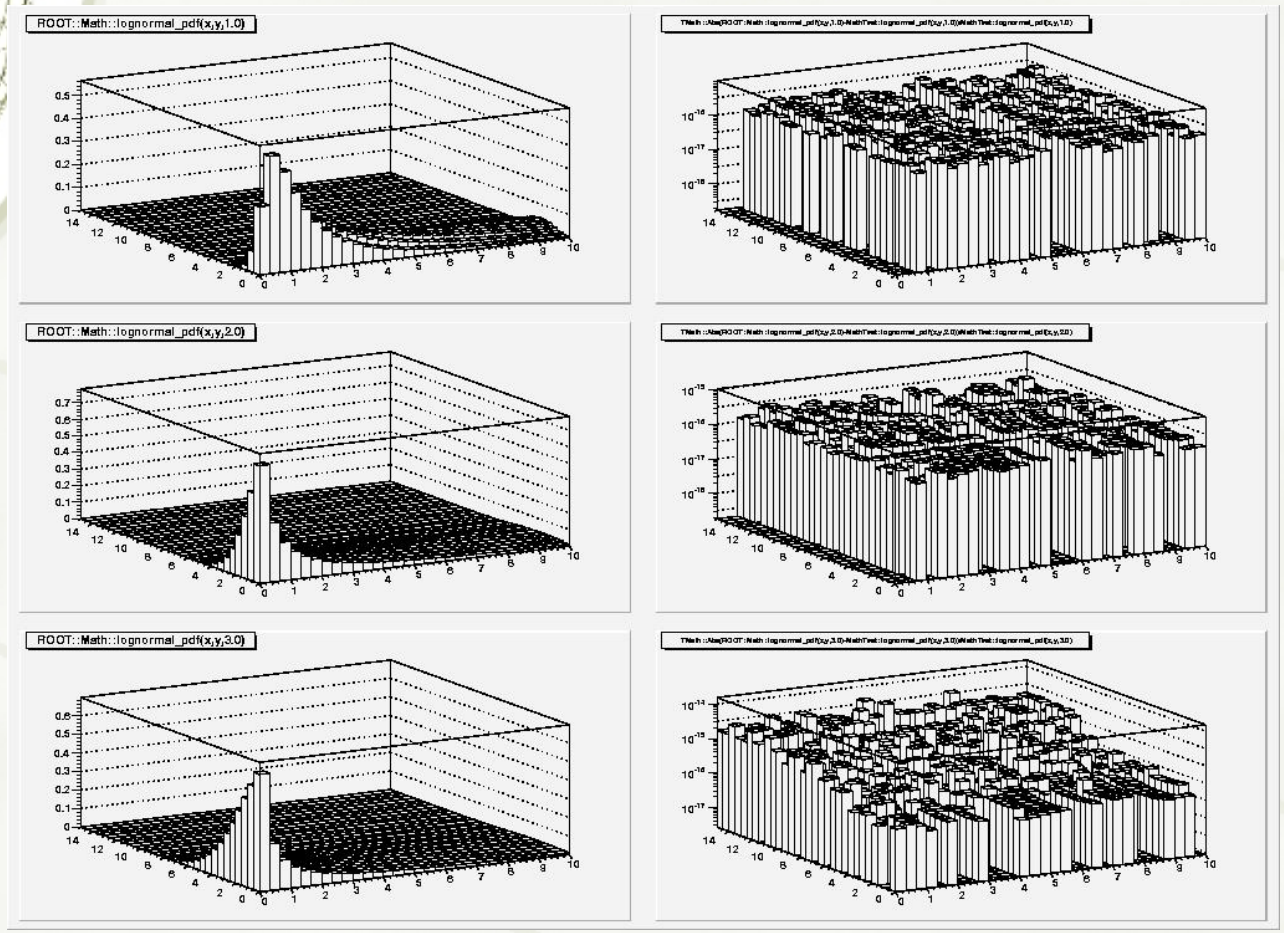


Statistical Functions

- Common functions used in statistics with a coherent naming scheme :
  - Probability density functions (pdf)
  - Cumulative distributions (lower tail and upper tail)
  - Inverse of cumulative distributions
- Examples:
  - `chisquared_pdf`
  - `chisquared_prob, chisquared_quant`
  - `chisquared_prob_inv, chisquared_quant_inv`
- Naming convention proposed for the C++ standard in N1668, but might change (to be followed closely)
  - http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1069.pdf

Statistical Functions Tests

σ =1.0



$\Delta \approx 10^{-16}$

σ =2.0

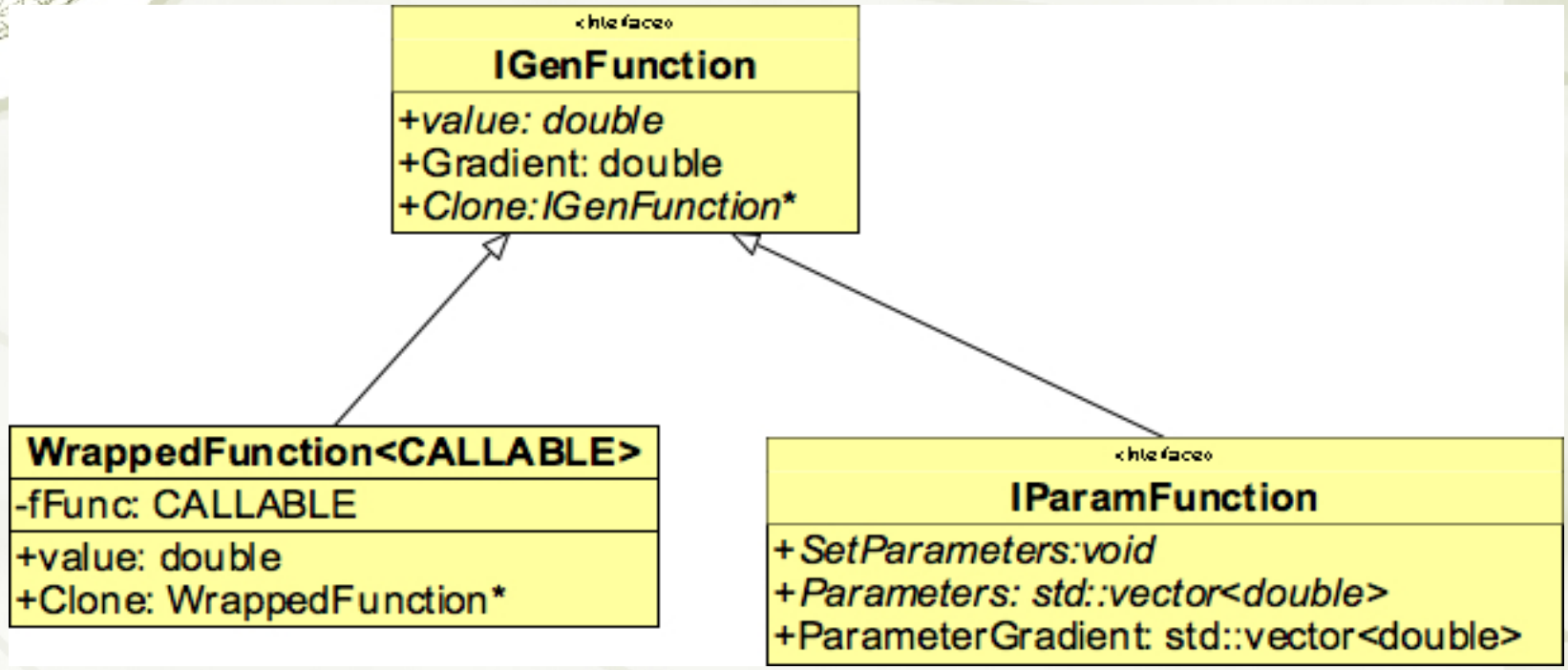
$\Delta \approx 10^{-16}$

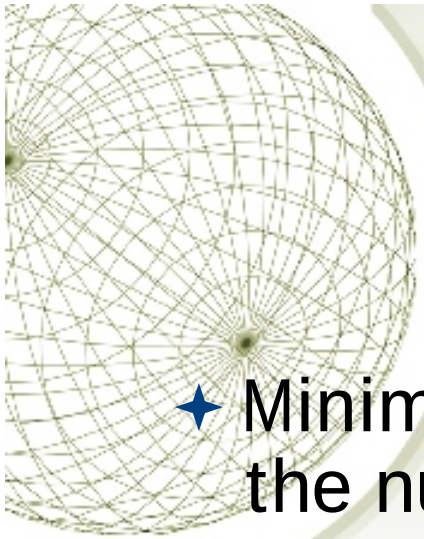
σ =3.0

$\Delta \approx 10^{-16}$

Lognormal PDF (left) and its Relative Difference Compared to Mathematica (right)

Function Interfaces





Function Interface

- Minimal interface for functions used by all the numerical algorithms: IGenFunction, ParamFunction, Polynomial (see previous presentations)
- class WrappedFunction<T> which wraps any C++ callable object (C free functions, functors, etc…)
- Reviewed by C++ experts — several of the recommendations implemented

Derivation

- Adaptive central difference algorithm using a 5-point rule
- Adaptive forward difference algorithm using a 4-point rule
- Adaptive backward difference algorithm using a 4-point rule

Derivation — an example of the overall design

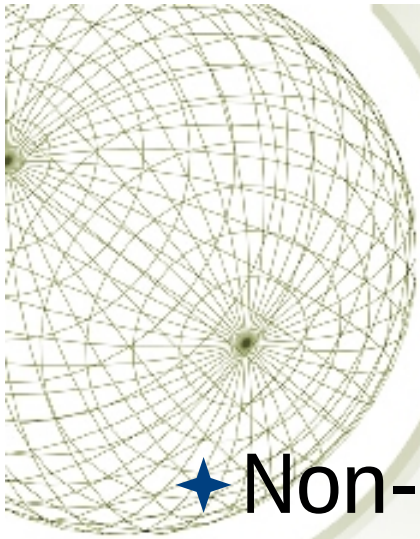
- Usage with function inheriting from IGenFunction:

```
ROOT::Math::Polynomial *f1 = new ROOT::Math::Polynomial(2);
…
ROOT::Math::Derivator *der = new ROOT::Math::Derivator(*f1);
double x0 = 2;
double result = der->EvalCentral(x0);
double error = der->Error();
```

Derivation — an example of the overall design, cont'd

- Usage with a function pointer:

```
double myfunc ( double x, void * ) {
  return std::pow( x, 1.5);
}

ROOT::Math::Derivator *der = new
ROOT::Math::Derivator(myfunc);
double x0 = 2;
double result = der->EvalCentral(x0);
```



Integration

- Non-adaptive, adaptive and adaptive singular (i.e. taking into account singularities) integration
- Different Gauss-Konrod rules can be selected
- Possibility to use infinite and semi-infinite ranges

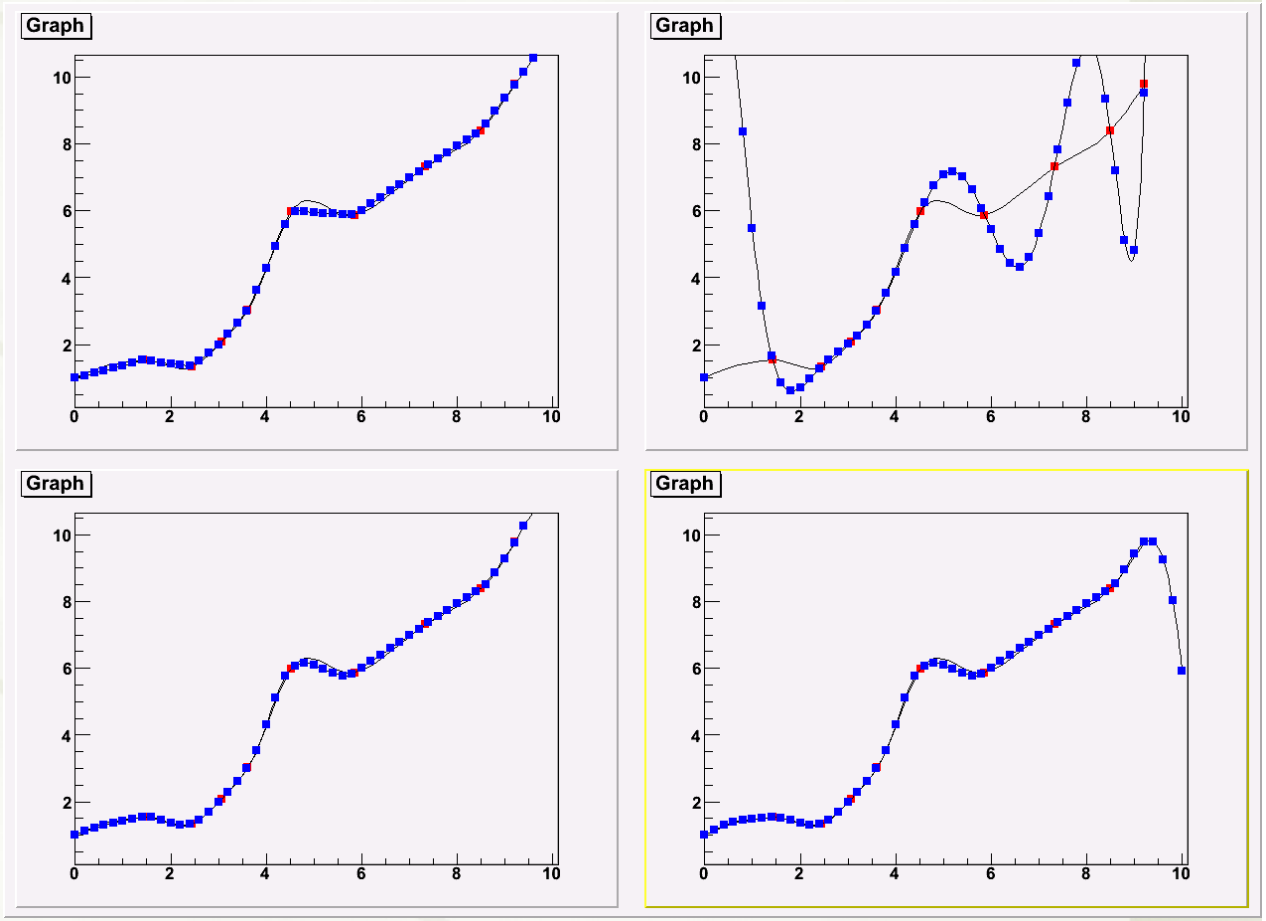
Integration Example

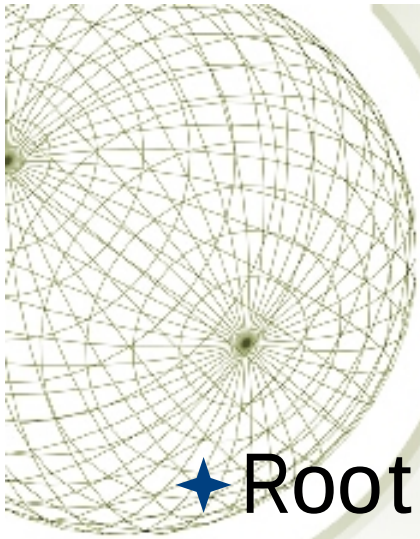
```
// user provided free C function
double myFunc ( double x) {… }

// wrap the function
ROOT::Math::WrappedFunction wFunc(myFunc);

// create integrator and integrate
ROOT::Math::Integrator ig(wFunc);
Double result = ig.Integral(a, b);
Double error = ig.Error();
```

- WrappedFunction can be replaced with any IGenFunction in the Integrator

Interpolation

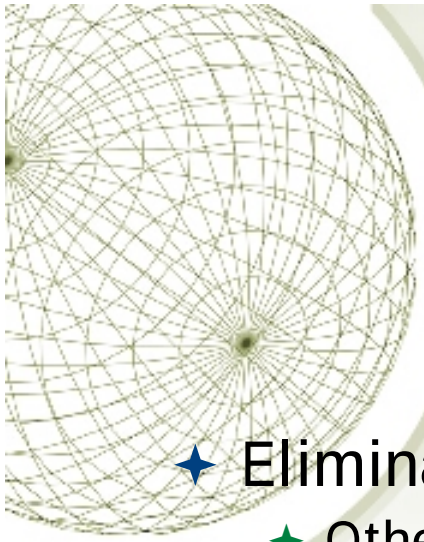
- Linear, polynomial, Akima and Akima periodic interpola-tions





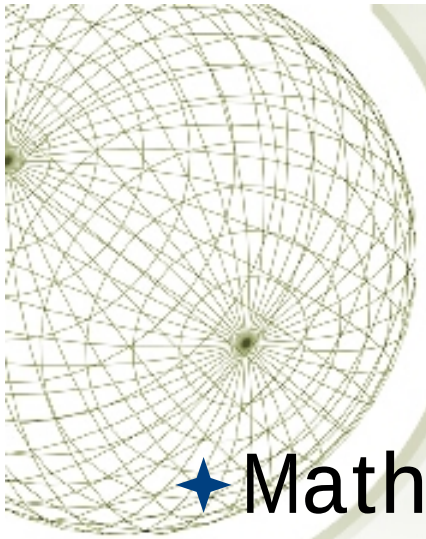
Root Finding

- Root finding of one dimensional functions
- Bracketing algorithms: bisection, false position, Brent-Dekker
- Polishing algorithms (derivatives): Newton, secant, Steffenson



Next Steps

- Eliminate duplication
  - Other parts of the ROOT should use mathcore/mathmore
  - Moving functionality from TMath into mathcore/mathmore
    (TMath will remain for backward compatibility)
- Implement incomplete gamma function in mathcore (for χ^2)
- A more detailed discussion is needed to finalize function interfaces (signatures)
- Prototype version of TF1 using algorithms from mathcore and implementing function interface
- Add algorithms for multi-dimensional functions
- New additions according to user requests (ex. FFT)



Conclusions

- Mathmore is available in ROOT 5.04/00 (`--enable-mathmore` switch in configure)
- Works on all supported platforms
- Documentation available at
  - http://seal.web.cern.ch/seal/MathLibs/MathMore-5_0_4/html/index.html