Обзор библиотек C++: статистика, оптимизация, анализ данных

Дорофеев Данила 674 гр

Краткие сведения о C++

C++ создан Бьёрном Страуструпом в 1980-х как усовершенствование C.

Принципы языка:

- Получить универсальный язык со статическими типами данных, эффективностью и переносимостью языка Си.
- Непосредственно и всесторонне поддерживать множество стилей программирования, в том числе процедурное программирование, абстракцию данных, объектно-ориентированное программирование и обобщённое программирование.
- Дать программисту свободу выбора, даже если это даст ему возможность выбирать неправильно.
- Максимально сохранить совместимость с Си, тем самым делая возможным лёгкий переход от программирования на Си.
- Избегать особенностей, которые зависят от платформы или не являются универсальными.
- Не требовать слишком усложнённой среды программирования.

Компиляторы C++

- Borland C++
- Microsoft Visual C++
  MSVC входит в состав MS Visual Studio
- GNU Compiler Collection
  GCC - стандартный компилятор для UNIX
- MinGW
  программный порт GCC под Microsoft Windows
- Intel C++ compiler
  оптимизирующий компилятор для x86, x86-64 и IA-64

Intel C++ compiler

Основные возможности:

- Высокоуровневая оптимизация
- Межпроцедурная оптимизация
- Автоматическое распараллеливание кода
- Векторизация
- Разделение циклов по нескольким нитям
- Профилирующая оптимизация

Библиотеки C++

- Boost (включая uBLAS)
- Intel® Math Kernel Library
- ALGLIB
- ROOT

- Boost - это собрание библиотек, расширяющих C++.
- Свободно распространяются по лицензии Boost Software License вместе с исходным кодом.
- Часть библиотек являются кандидатами на включение в следующий стандарт C++.

http://www.boost.org/

Некоторые библиотеки

- Алгоритмы
- Многопоточное программирование
- Юнит-тестирование
- Обобщённое программирование
- Графы
- Итераторы
- **Математические и числовые алгоритмы**
- Работа с памятью
- Синтаксический и лексический разбор
- Метапрограммирование на основе препроцессора
- «Умные указатели»
- Обработка строк и текста
- Метапрограммирование на основе шаблонов

Статистика

Распределение вероятностей

- Стьюдента <students_t.hpp>
- χ^2 (Хи-квадрат) <chi_squared.hpp>
- Биномиальное <binomial.hpp>
- Нормальное <normal.hpp>
- Пуассона <poisson.hpp>
- Бернулли <bernoulli.hpp>

Распределение вероятностей

Функция плотности вероятности
probability density function

- p = pdf(normal_distribution, x);

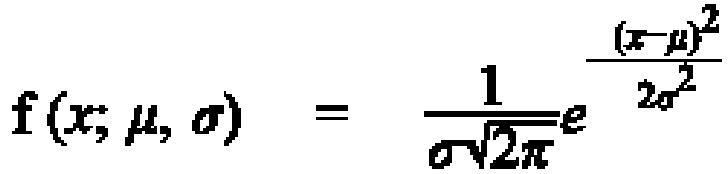
Кумулятивная функция распределения
Cumulative distribution function

- c = cdf(binomial_distribution, x);

Квантиль

- q = quantile(chi_squared_distribution, p);

Нормальное распределение



$$f(x; \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{(x-\mu)^2}{2\sigma^2}}$$

```
#include <boost/math/distributions/normal.hpp>
using namespace boost::math;

double mean_life = 1100.;
double life_standard_deviation = 100.;
normal bulbs(mean_life, life_standard_deviation);
double expected_life = 1000.;

cout << "Fraction of bulbs that will last at best (<=) " // P(X <= 1000)
  << expected_life << " is "<< cdf(bulbs, expected_life) << endl;
cout << "Fraction of bulbs that will last at least (>) " // P(X > 1000)
  << expected_life << " is "<< cdf(complement(bulbs, expected_life)) <<
   endl;
double min_life = 900;
double max_life = 1200;
cout << "Fraction of bulbs that will last between "
  << min_life << " and " << max_life << " is "
  << cdf(bulbs, max_life)  // P(X <= 1200)
   - cdf(bulbs, min_life) << endl; // P(X <= 900)
```

Линейная алгебра

Принятые определения:

`A, B, C`	Матрицы
`u, v, w`	вектора
`i, j, k`	целые числа
`t, t1, t2`	скаляры
`r, r1, r2`	интервалы, range(0, 3)

Стандартные операции:

```
C = A + B; C = A - B; C = -A;
w = u + v; w = u - v; w = -u;
C = t * A; C = A * t; C = A / t;
w = t * u; w = u * t; w = u / t;
```

Линейная алгебра

boost C++ LIBRARIES

Вычисляемые присвоения

```
C += A; C -= A;
w += u; w -= u;
C *= t; C /= t;
w *= t; w /= t;
```

Скалярное, векторное и другие произведения

```
t = inner_prod(u, v);
C = outer_prod(u, v);
w = prod(A, u); w = prod(u, A); w = prec_prod(A, u);
w = prec_prod(u, A);
C = prod(A, B); C = prec_prod(A, B);
w = element_prod(u, v); w = element_div(u, v);
C = element_prod(A, B); C = element_div(A, B);
```

Многомерные массивы


```
#include "boost/multi_array.hpp"
#include <cassert>

// Create a 3D array that is 3 x 4 x 2
  typedef boost::multi_array<double, 3> array_type;
  typedef array_type::index index;
  array_type A(boost::extents[3][4][2]);

  // Assign values to the elements
  int values = 0;
  for(index i = 0; i != 3; ++i)
    for(index j = 0; j != 4; ++j)
      for(index k = 0; k != 2; ++k)
        A[i][j][k] = values++;
```

Intel® Math Kernel Library

- Библиотека Intel® Math Kernel Library (Intel® MKL) обеспечивает выполнение высокооптимизированных многопоточных математических операций для научных, инженерных и финансовых приложений, требующих наибольшей производительности. В неё включены различные математические функции, в том числе библиотеки BLAS, LAPACK, ScaLAPACK.

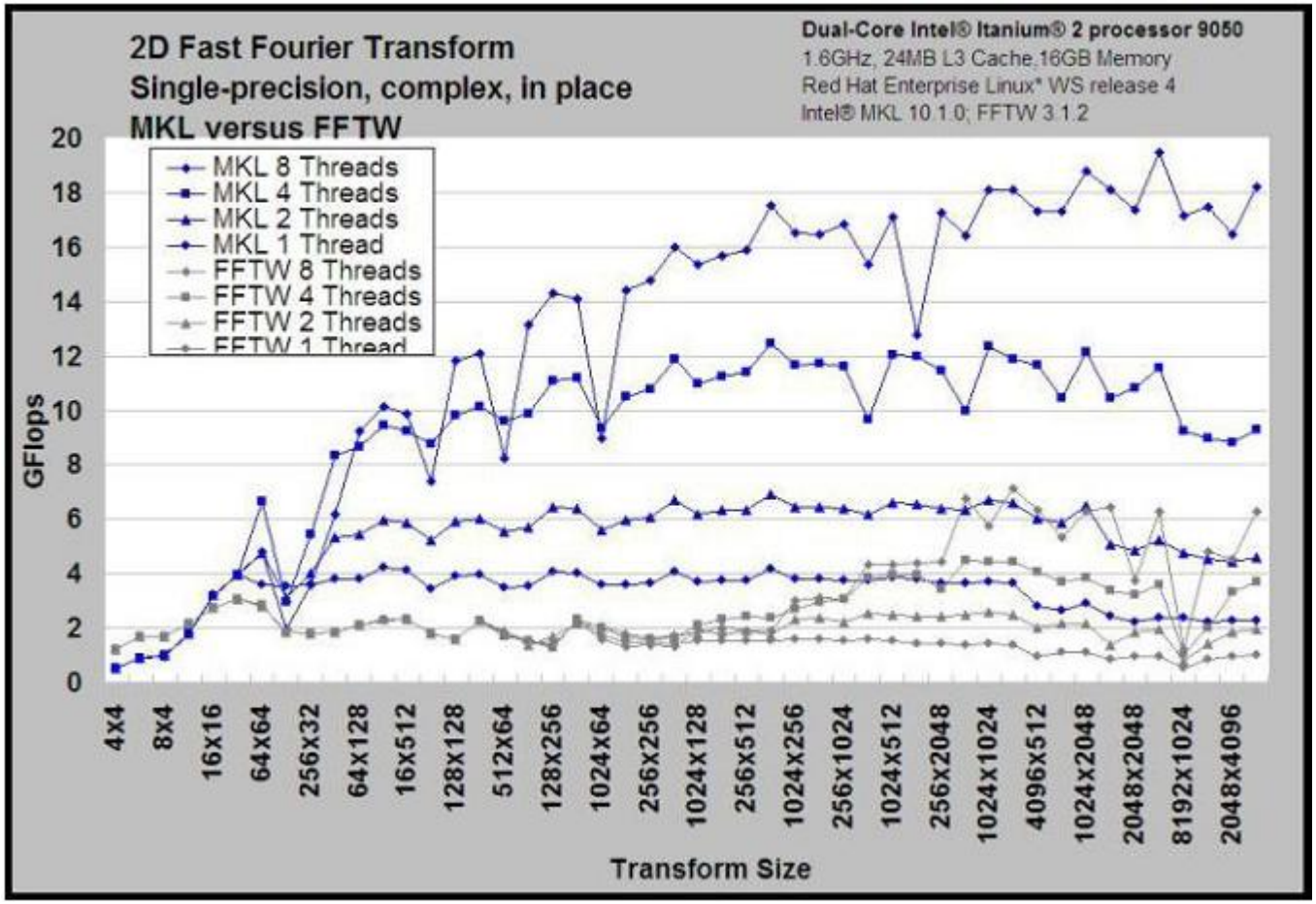
Стоимость Intel® Math Kernel Library for Windows*

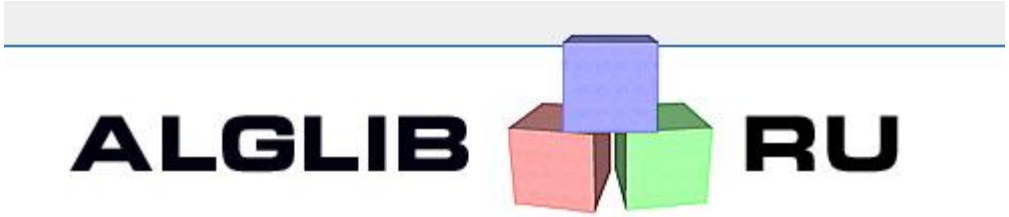
$399 - Full Product $160 - Support Renewal

MKL содержит следующие библиотеки:

- - BLAS
- - Sparse BLAS
- - LAPACK
- - PBLAS
- - ScaLAPACK
- - Sparse Solver routines
- - Vector Mathematical Library functions
- - Vector Statistical Library functions
- - Fourier Transform functions (FFT)
- - Cluster FFT
- - Trigonometric Transform routines
- - Poisson, Laplace, and Helmholtz Solver routines
- - Optimization (Trust-Region) Solver routines
- - GMP* arithmetic functions

Производительность пакета





Что такое ALGLIB

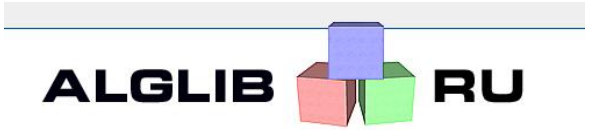
- ALGLIB – это многоязыковая коллекция алгоритмов для решения проблем в области численного анализа и обработки данных. ALGLIB распространяется под лицензией GPL.

Основные достоинства ALGLIB

- Поддержка нескольких языков программирования. (C++, C#, MPFR, FreePascal, Delphi, VBA)
- 100% реализация функций пакета в рамках любого из поддерживаемых языков.
- Переносимость и простота компиляции. Собирается в практически любой среде под практически любым компилятором.
- Приемлемая производительность. ALGLIB не может соревноваться с программными пакетами, использующими низкоуровневую оптимизацию (например, MKL).
- Богатая документация. Документация содержит не только описание подпрограмм и примеры, но и обсуждение алгоритмов, их сильных и слабых сторон.

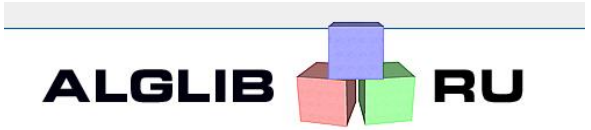
http://alglib.sources.ru/

Библиотеки ALGLIB



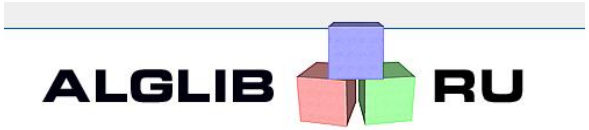
- Решение обыкновенных дифференциальных уравнений
- Линейные уравнения
- Операции с матрицами и векторами
- Нахождение собственных значений и векторов
- Разреженные матрицы: итеративные алгоритмы
- Численное интегрирование
- Интерполяция, аппроксимация и численное дифференцирование
- Одномерная и многомерная оптимизация
- БПФ, свертка, корреляция
- **Статистика: общие алгоритмы**
- **Проверка гипотез**
- **Классификация, регрессия, кластеризация, работа с данными**

Анализ данных



- Линейная регрессия
- Множественная логит-регрессия
- Нейронные сети
- Ансамбли нейронных сетей
- Лес деревьев решений
- Линейный дискриминантный анализ
- Метод главных компонент
- Кластеризация алгоритмом k-means++

Алгоритмы описательной статистики



Функция определения моментов: среденее значение, среднее отклонение, коэффициент асимметрии, эксцесс распределения.

```
/*************************
Input parameters:
    X       -   sample. Array with whose indexes range within [0..N-1]
    N       -   sample size.

Output parameters:
    Mean    -   mean.
    Variance-   variance.
    Skewness-   skewness (if variance<>0; zero otherwise).
    Kurtosis-   kurtosis (if variance<>0; zero otherwise).
*************************/
void calculatemoments(const ap::real_1d_array& x,
                      int n,
                      double& mean,
                      double& variance,
                      double& skewness,
                      double& kurtosis);
```

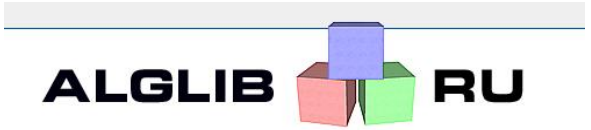
Логит-регрессия

ALGLIB RU

Описание функции нелинейной регрессии

```
/*************************************************************************
Weighted nonlinear least squares fitting using gradient/Hessian.

Nonlinear task min(F(c)) is solved, where

    F(c) = (w[0]*(f(x[0],c)-y[0]))^2 + ... + (w[n-1]*(f(x[n-1],c)-y[n-1]))^2,

    * N is a number of points,
    * M is a dimension of a space points belong to,
    * K is a dimension of a space of parameters being fitted,
    * w is an N-dimensional vector of weight coefficients,
    * x is a set of N points, each of them is an M-dimensional vector,
    * c is a K-dimensional vector of parameters being fitted

This subroutine uses f(x[i],c), its gradient and its Hessian.

See LSFitNonlinearWFG() subroutine for information about function
parameters.

*************************************************************************/
```

Логит-регрессия




```
// Fitting 0.5(1+cos(x)) on [-pi,+pi] with exp(-alpha*x^2)
// Solve

    lsfitnonlinearfg(x, y, c, n, m, k, true, state);
    lsfitnonlinearsetcond(state, epsf, epsx, maxits);
    while(lsfitnonlineariteration(state))
    {
        if( state.needf )
        {

            //
            // F(x) = Exp(-alpha*x^2)
            //
            state.f = exp(-state.c(0)*ap::sqr(state.x(0)));
        }
        if( state.needfg )
        {
          //
            // F(x)       = Exp(-alpha*x^2)
            // dF/dAlpha = (-x^2)*Exp(-alpha*x^2)
            //
            state.f = exp(-state.c(0)*ap::sqr(state.x(0)));
            state.g(0) = -ap::sqr(state.x(0))*state.f;
        }
    }
    lsfitnonlinearresults(state, info, c, rep);
```

Нейронные сети

Работа с нейронными сетями

Нейронная сеть в ALGLIB представлена структурой MultiLayerPerceptron. Работа с нейронными сетями осуществляется в такой последовательности:

1. Выбор архитектуры и инициализация структуры при помощи соответствующей функции.
2. Обучение нейронной сети при помощи одного из алгоритмов.
3. Использование обученной сети (применение к входным данным, сериализация и т.д.).

Нейронные сети

ALGLIB RU

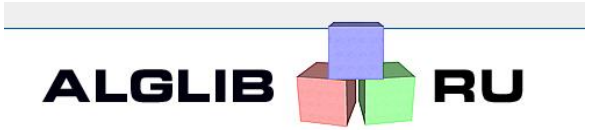
Алгоритмы обучения

1. L-BFGS алгоритм (limited memory BFGS), квази-Ньютоновский метод с трудоемкостью итерации, линейной по количеству весовых коэффициентов WCount и размеру обучающего множества NPoints, и умеренными требованиями к дополнительной памяти - O(WCount).

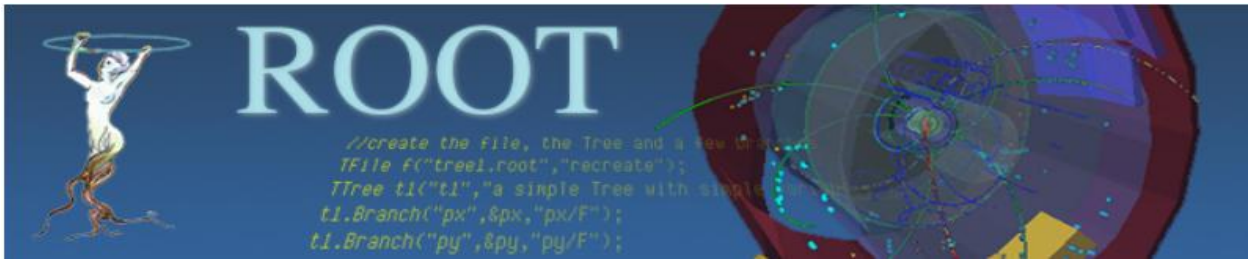
2. Модифицированный метод Левенберга-Марквардта, использующий точный гессиан функции ошибки (НЕ линеаризованную аппроксимацию). На задачах малой и средней размерности (до нескольких сотен весовых коэффициентов) этот алгоритм часто оказывается быстрее L-BFGS, но его главное достоинство - даже не скорость работы, а то, что он вообще не нуждается в указании критериев останова. Трудоемкость итерации равна O(NPoints·WCount 2), затраты памяти - O(WCount 2).

3. Метод раннего останова. Этот метод используется в составе одного из алгоритмов конструирования ансамблей нейронных сетей.

Обучение нейронного классификатора




```
multilayerperceptron net;
ap::real_1d_array  x, y;

// classification task with 2 inputs and 3 classes.
// network weights are initialized with small random values.

mlpcreatec0(2, 3, net);
x.setlength(2);
y.setlength(3);
x(0) = ap::randomreal()-0.5;
x(1) = ap::randomreal()-0.5;
mlpprocess(net, x, y);

// output results

printf("Classification task\n");
printf("IN[0]  = %5.2lf\n",   double(x(0)));
printf("IN[1]  = %5.2lf\n",   double(x(1)));
printf("Prob(Class=0|IN) = %5.2lf\n",   double(y(0)));
printf("Prob(Class=1|IN) = %5.2lf\n",   double(y(1)));
printf("Prob(Class=2|IN) = %5.2lf\n",   double(y(2)));
```



Что такое ROOT?

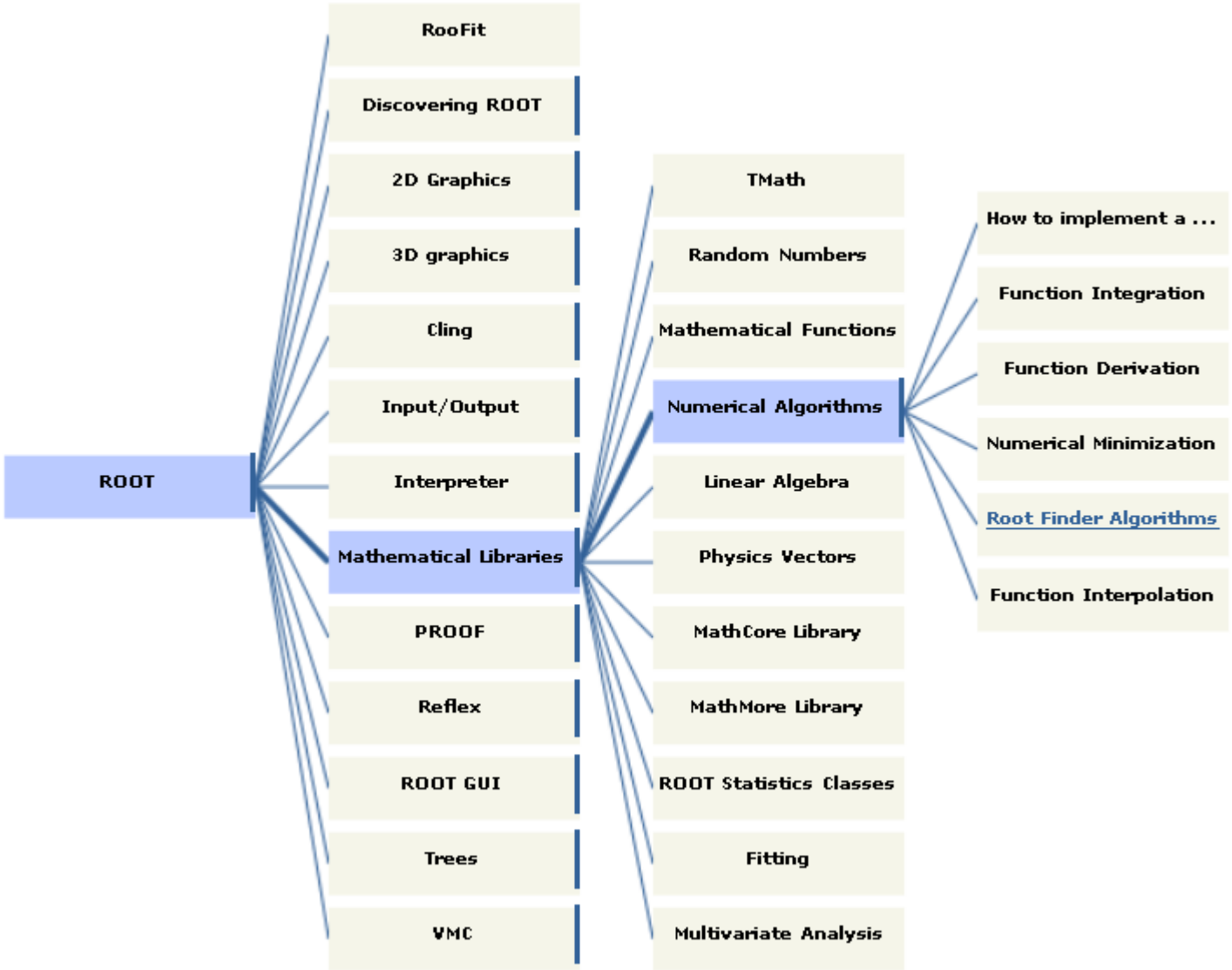
- ROOT - предоставляет набор библиотек необходимыми для эффективной обработки и анализа больших объемов данных. Данные определяются как набор объектов и методов для прямого доступа к конкретным атрибутам, без загрузки всего массива данных.

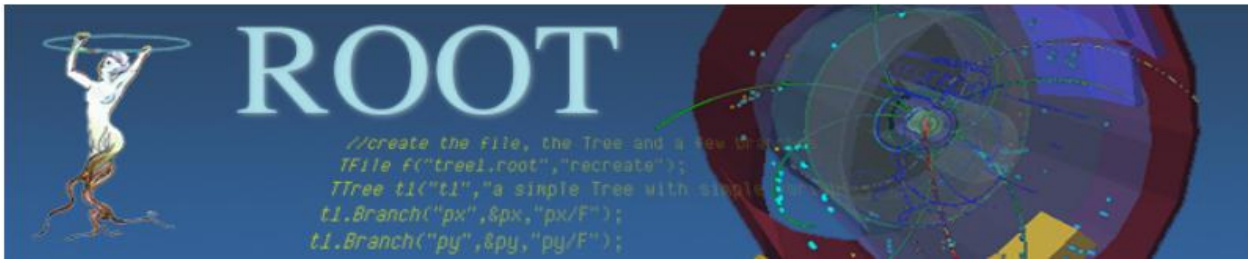
- Разрабатывается сотрудникам CERN для экспериментов в области физики высоких энергий.

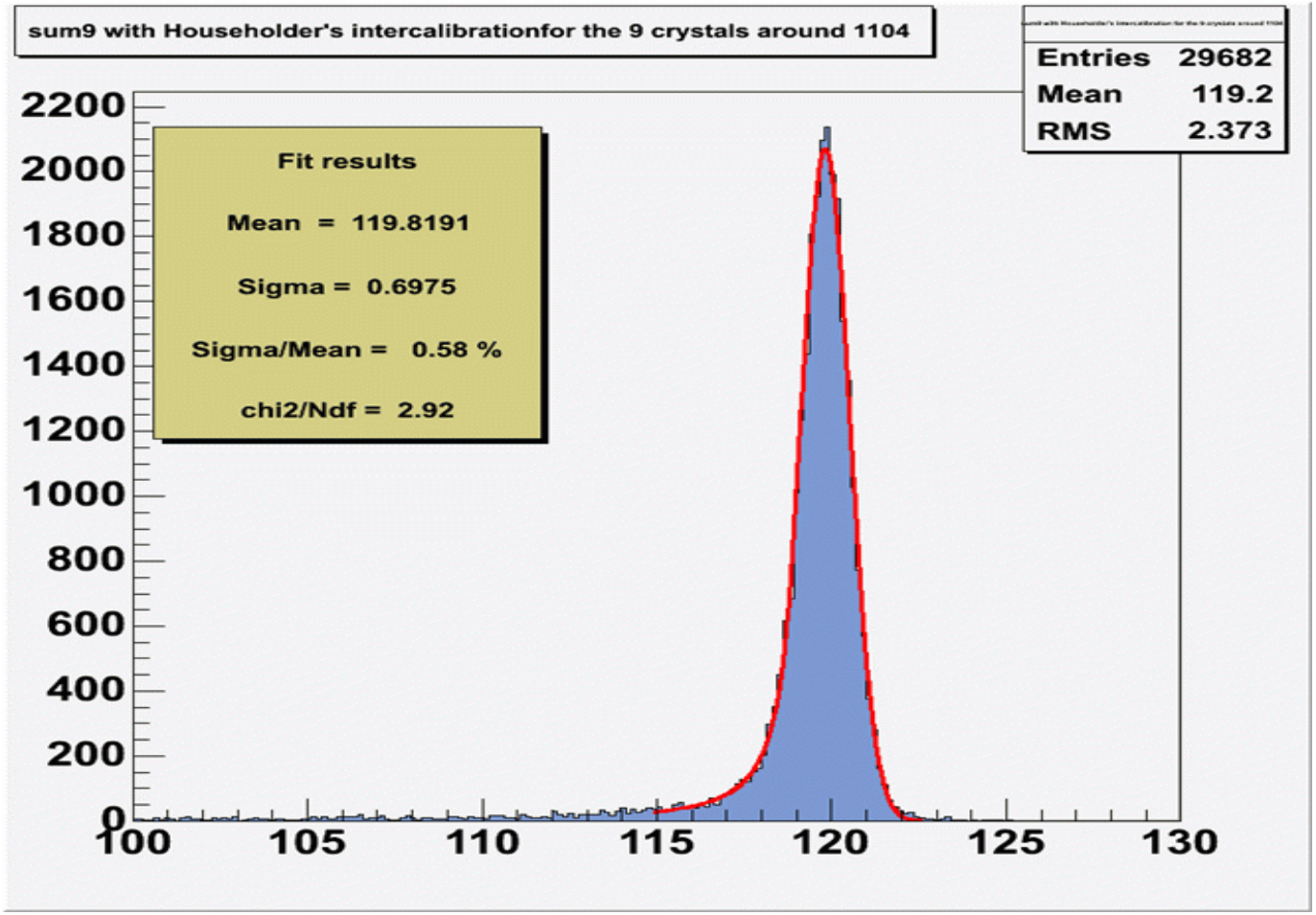
- Бесплатна, кросплатформенна. Доступна для скачивания.

root.cern.ch

Библиотеки ROOT




ROOT
//create the file, the Tree and a few br
TFile f("tree1.root","recreate");
TTree t1("t1","a simple Tree with simple
t1.Branch("px",&px,"px/F");
t1.Branch("py",&py,"py/F");


sum9 with Householder's intercalibrationfor the 9 crystals around 1104
Entries 29682
Mean 119.2
RMS 2.373
Fit results
Mean = 119.8191
Sigma = 0.6975
Sigma/Mean = 0.58 %
chi2/Ndf = 2.92
2200
2000
1800
1600
1400
1200
1000
800
600
400
200
0
100
105
110
115
120
125
130

Использование C++ для анализ данных

Преимущества

- В среде C++ существует большое количество сторонних библиотек для анализа данных.
- Многие из них бесплатны.
- Программы на C++ работают быстрее интерпретируемых языков.
- Возможность гибкой подстройки параметров алгоритма.

Использование C++ для анализ данных

Недостатки

- Большой размер кода. 1 строка кода MATLAB записывается 5-8 строками на C++.
- Сложность и избыточность, из-за которых C++ трудно изучать.
- Синтаксис, провоцирующий ошибки.

Выводы

Для проведения вычислительного эксперимента разумно пользоваться высокоуровневыми языками программирования, такими как Maple, MATLAB и др.

Для коммерческой реализации продукта можно воспользоваться широкими возможностями языка C++.

Спасибо за внимание