

Моделирование и обработка результатов измерений

Смирнов Сергей Юрьевич

ИЯФиТ, кафедра 40

sysmirnov@mephi.ru

Моделирование и обработка результатов измерений

Практические работы

Тема 1: «Метод наименьших квадратов»

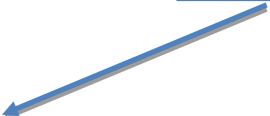
- 1: линейный МНК с графическим изображением полученной функции
- 2: линейный МНК с ошибками по оси Y в каждой точке и с графиком функции
- 3: квадратичный МНК без учета ошибок, с графиком функции
- 4: фитирование экспериментальных распределений функцией Гаусса

Тема 2: «Моделирование и обработка результатов по распаду π^0 -мезона на 2 γ -кванта»

- 5: моделирование распада $\pi^0 \rightarrow \gamma + \gamma$ в системе покоя π^0 -мезона
 - 6: преобразование кинематических характеристик вторичных частиц (γ -квантов) в лабораторную систему отсчета, формулы Лоренц-преобразования
 - 7: моделирование детектора γ -квантов и запись модельных сигналов детектора во внешний файл
 - 8: анализ экспериментальных данных по распаду π^0 -мезона. Чтение файла из работы № 7 и проверка кинематических параметров вторичных частиц
 - 9: построение массового спектра системы двух γ -квантов и восстановление массы родительской частицы
- Позапное развитие программы с математической моделью распада частицы и детектирования продуктов распада
- Создание второй программы, осуществляющей анализ модельных данных

Рабочая платформа – Linux

Scientific Linux CERN (SLC) vers. 6.10

- Серверы:
 - ui02.lxfarm.mephi.ru (основной)
 - ui03.lxfarm.mephi.ru (более свежая версия Linux — CentOS 7.9)
 - pm02.lxfarm.mephi.ru
 - pm04.lxfarm.mephi.ru
- Создание файла с текстом программы:
 - редакторы: [vi](#), [emacs](#), [pico](#), [nano](#) и многие другие
- Компиляция:
`g++ file.cxx` или `g++ file.cxx -o file.exe`
- Запуск на выполнение:
`./a.out` или `./file.exe`

Допустимые в Unix расширения имени файла с текстом программы на C++:

`.C .cc .cpp .cxx`

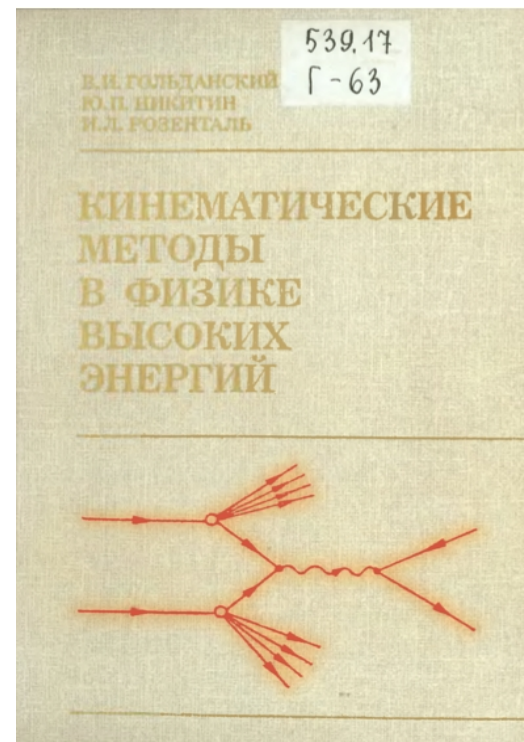
Рекомендуемая литература

ROOT:

- <https://root.cern/primer/> (англ.)
- <https://root.cern/manual/> (англ.)
- <http://nuclphys.sinp.msu.ru/books/b/ROOT.htm> (рус.)

Кинематические методы:

- В.И.Гольданский, Ю.П.Никитин, И.Л.Розенталь
Москва “Наука” 1987 г.



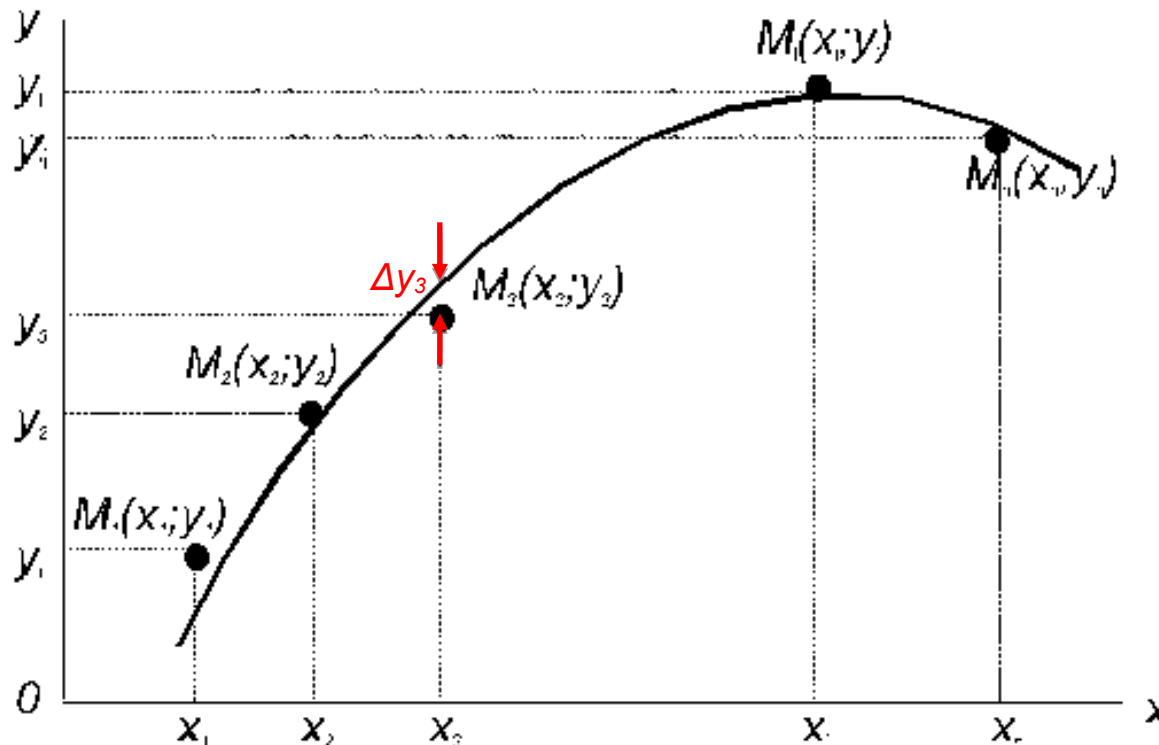
Практическая работа №1

Метод наименьших квадратов

Фитирование наблюдаемых точек линейной функцией

Метод наименьших квадратов – общий метод построения оценок неизвестных параметров модели (теории) случайного явления как значений параметров, минимизирующих суммы квадратов отклонений, т.е. разностей между наблюдениями и их теоретическими величинами (как функциями от неизвестных искомых параметров).

Метод применяется также при аппроксимации функций.



$M_i(x_i, y_i)$ – наблюдаемые точки и их координаты

Δy_i – отклонение i -ой наблюдаемой точки от теоретической кривой

Модель с линейной функцией:

$$y = f(x) = a \cdot x + b$$

Суть метода наименьших квадратов:

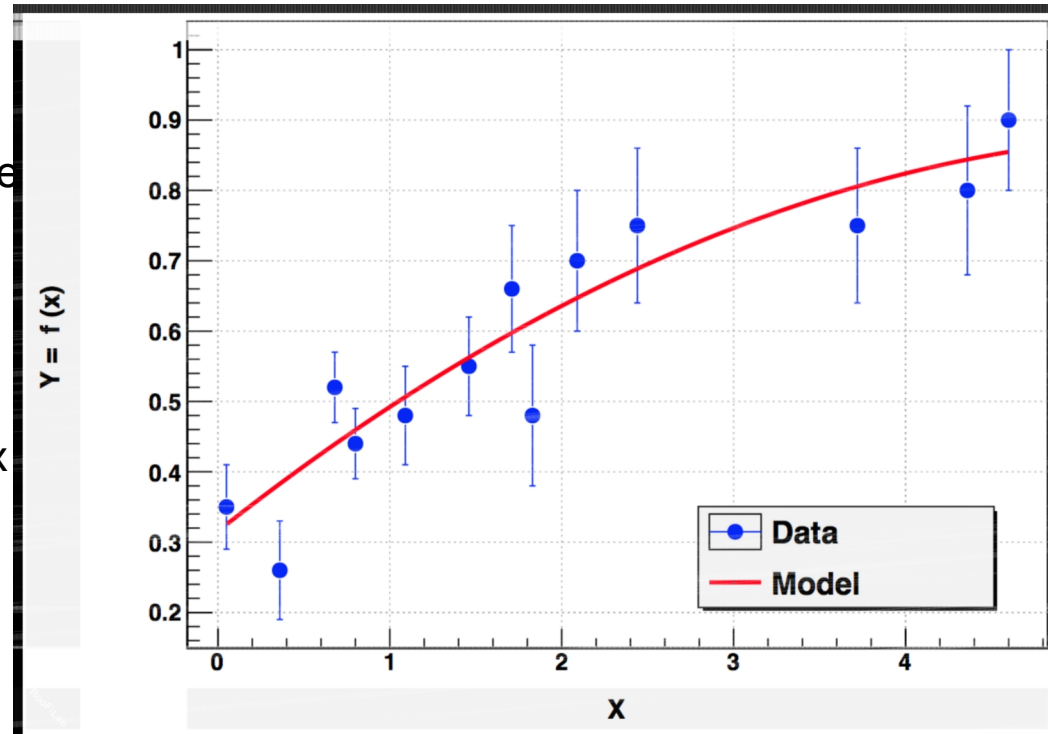
$$\sum (y_i - f(x_i))^2 = \min$$

Практическая работа №1

Метод наименьших квадратов Фитирование наблюдаемых точек линейной функцией

Задание:

- исходя из метода наименьших квадратов, вывести аналитические формулы для вычисления неизвестных параметров линейной функции a , b по известным координатам наблюдаемых точек
- написать программу, в которую с клавиатуры вводится количество наблюдаемых точек и их координаты, вычисляются и печатаются найденные параметры линейной функции
- для визуализации полученного результата программа с помощью графического пакета **Root** должна нарисовать введенные точки и поверх них построить получившуюся линейную функцию



Слайды с подсказками по программному
пакету ROOT

Вариант 1: работа внутри программной оболочки Root с помощью интерпретатора CINT/Cling

```
ui02.lxfarm> root
root [0] .x testMNK.C - запуск скрипта на выполнение
root [1] .q          - выход из программной оболочки Root
```

Вариант 2: создание собственных исполняемых программ с подключением библиотек из пакета Root

```
ui02.lxfarm> g++ -o test1.exe test1.cpp `root-config --cflags --libs`
ui02.lxfarm> ./test1.exe
```

testMNK.C

```
#include "TCanvas.h"
#include "TGraph.h"
#include "TGraphErrors.h"
void testMNK()
{
    TCanvas *canv1 = new TCanvas("canv1", "LSQ example", 0, 0, 1000, 700);
    float x1[4] = {1, 2, 3, 4};
    float y1[4] = {1, 2.1, 2.95, 4.15};
    float ex[4] = {0, 0, 0, 0};
    float ey[4] = {0.2, 0.3, 0.25, 0.2};
    TGraphErrors *gr1 = new TGraphErrors(4, x1, y1, ex, ey);
    gr1->SetMarkerStyle(21);
    gr1->Draw("AP");
    float x2[10], y2[10];
    for(int i=0; i<10; i++)
    { x2[i] = i*0.5;
      y2[i] = x2[i];
    }
    TGraph *gr2 = new TGraph(10, x2, y2);
    gr2->Draw("Lsame");
    return;
}
```

```
#include "TApplication.h"
int main(int argc, char** argv)
{
    TApplication App("App", &argc, argv);
    void testMNK(); //function prototype
    testMNK();
    App.Run();
    return 0;
}
```

test1.cpp

Графики

- Работа с графиками обеспечивается классом TGraph
- Для создания графика нужно определить два массива, содержащих n значений абсцисс и ординат точек.
- Пример. График функции $y(x)=10*\sin(x+0.2)$

```
Int_t n = 20;  
Double_t x[n], y[n];  
for (Int_t i=0; i<n; i++) {  
    x[i] = i*0.1;  
    y[i] = 10*sin(x[i]+0.2);  
}  
TGraph *gr1 = new TGraph (n, x, y);
```

- Рисование графика
 - `gr1->Draw ("ACP")`

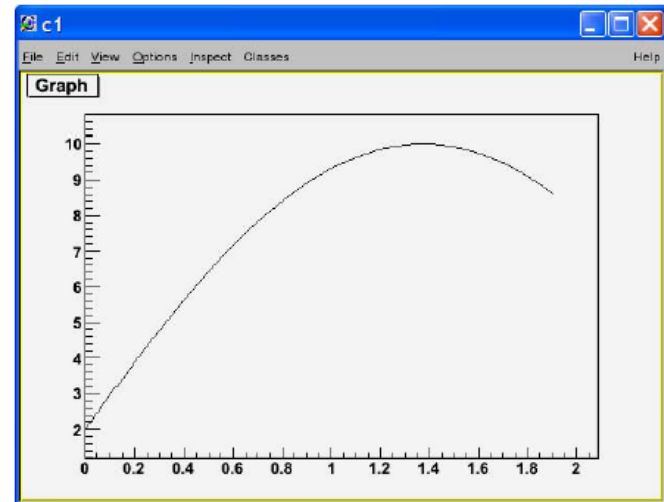
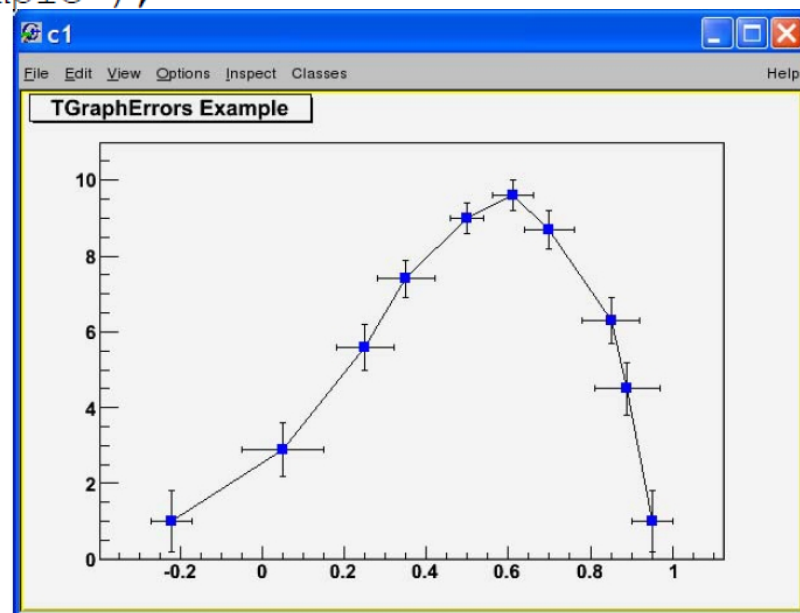


График с погрешностями

- Для графиков с погрешностями используется класс TGraphErrors
- Пример скрипта

```
{  
Int_t n = 10;  
Float_t x[n] = {-.22, .05, .25, .35, .5, .61, .7, .85, .89, .95};  
Float_t y[n] = {1, 2.9, 5.6, 7.4, 9, 9.6, 8.7, 6.3, 4.5, 1};  
Float_t ex[n] = {.05, .1, .07, .07, .04, .05, .06, .07, .08, .05};  
Float_t ey[n] = {.8, .7, .6, .5, .4, .4, .5, .6, .7, .8};  
TGraphErrors *gr = new TGraphErrors(n, x, y, ex, ey);  
gr->SetTitle("TGraphErrors Example");  
gr->SetMarkerColor(4);  
gr->SetMarkerStyle(21);  
gr->Draw("ALP");  
}
```



Пример построения графика по точкам

файл `test_graph2d.C`

```
// This script can be run from interactive Root:
// [ ] .x test_graph2d.C
//
// or it can be compiled outside Root CINT
// > g++ -o test_graph2d.exe test_graph2d.C `root-config --cflags --
libs`
//
// and then run
// > ./test_graph2d.exe
//

#include "TCanvas.h"
#include "TGraph2D.h"

// --- for standalone compilation only
#ifdef __CINT__
#include "TApplication.h"

int main(int argc, char** argv)
{
    TApplication App("App",&argc, argv);
    void test_graph2d(); //function prototype
    test_graph2d();
    App.Run();
    return 0;
}
#endif
// --- end of standalone compilation block
```

```
void test_graph2d()
{
    Int_t nbin=10;
    Double_t xmin,xmax,x,dx;
    Double_t ymin,ymax,y,dy;
    Double_t zmin,zmax,z;
    TCanvas *canv1 = new TCanvas("canv1","Graph2d
example",0,0,1000,700);
    xmin=0; xmax=200;
    ymin=0; ymax=100;
    zmin=0; TGraph2D* dt = new TGraph2D(nbin);
    dx=(xmax-xmin)/nbin;
    dy=(ymax-ymin)/nbin;
    Int_t k = 0;
    for(Int_t i=0;i<nbin;i++)
    {
        x = xmin + dx*i;
        for(Int_t j=0;j<nbin;j++)
        {
            y = ymin + dy*j;
            z = 0.02*x*x + 2*y - 100;
            if(z>zmin)
            {
                dt->SetPoint(k,x,y,z);
                k++;
            }
        }
    }
    dt->SetMinimum(zmin);
    dt->Draw("surf1");
}
```