Моделирование и обработка результатов измерений

Лабораторные работы

Тема 1: «Метод наименьших квадратов»

- М1: линейный МНК с графическим изображением полученной функции в программе ROOT
- М2: линейный МНК с ошибками по *Y* в каждой точке и с графиком функции
- М3: квадратичный МНК без ошибок, с графиком функции

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

- М4: моделирование распада $\pi^0 \rightarrow \gamma+\gamma$ в системе покоя π^0-мезона
- М5: преобразование кинематических характеристик вторичных частиц (γ-квантов) в лабораторную систему отсчета, формулы Лоренц-преобразования
- М6: моделирование детектора γ-квантов и запись модельных сигналов детектора во внешний файл
- М7: новая программа: анализ экспериментальных данных по распаду π^0-мезона. Чтение файла из работы № М6 и проверка кинематических параметров вторичных частиц
- М8: построение массового спектра системы двух γ-квантов и восстановление массы родительской частицы

Рабочая платформа - Linux

Scientific Linux CERN (SLC) v.6.10

- Серверы:
  - ui02.lxfarm.mephi.ru (основной)
  - pm02.lxfarm.mephi.ru (запасной)
  - pm03.lxfarm.mephi.ru (запасной)
- Создание файла с текстом программы
  - редакторы: vi, emacs, pico, nano
- Компиляция
  g++ file.cxx
- Запуск на выполнение
  ./a.out

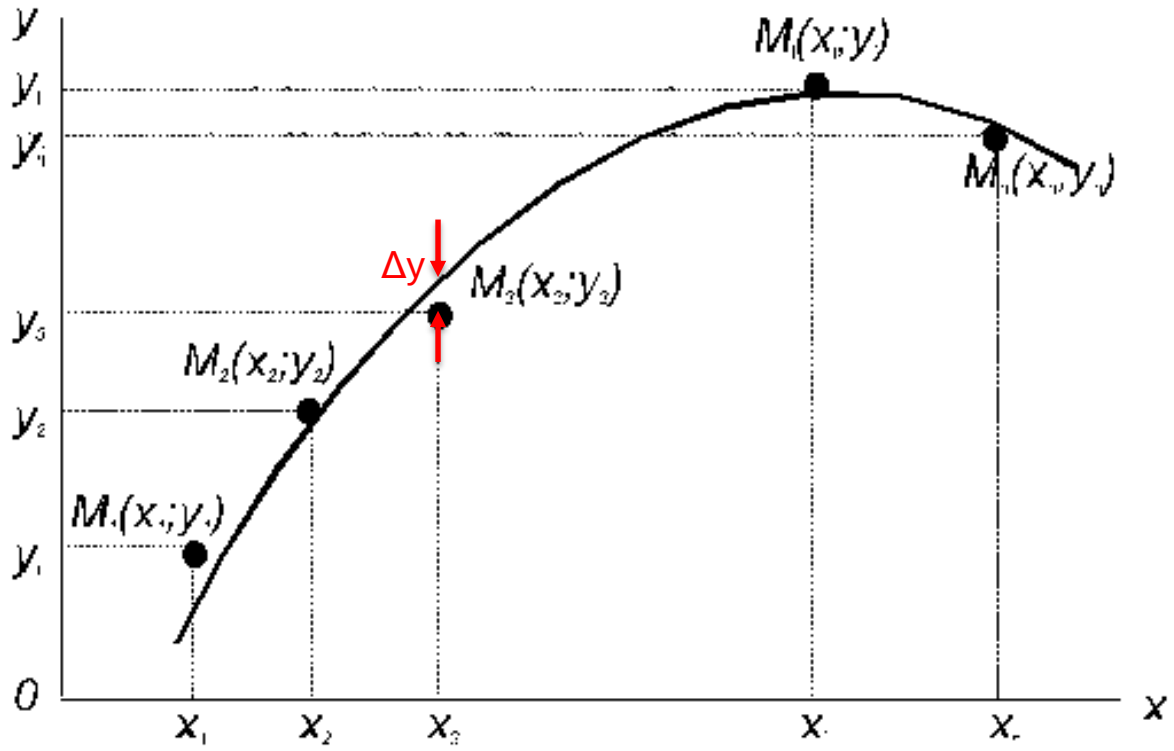
- Допустимые в Unix расширения имени файла с текстом программы на C++:
  .C .cc .cpp .cxx

Лабораторная работа М1

Метод наименьших квадратов. Фитирование линейной функцией

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

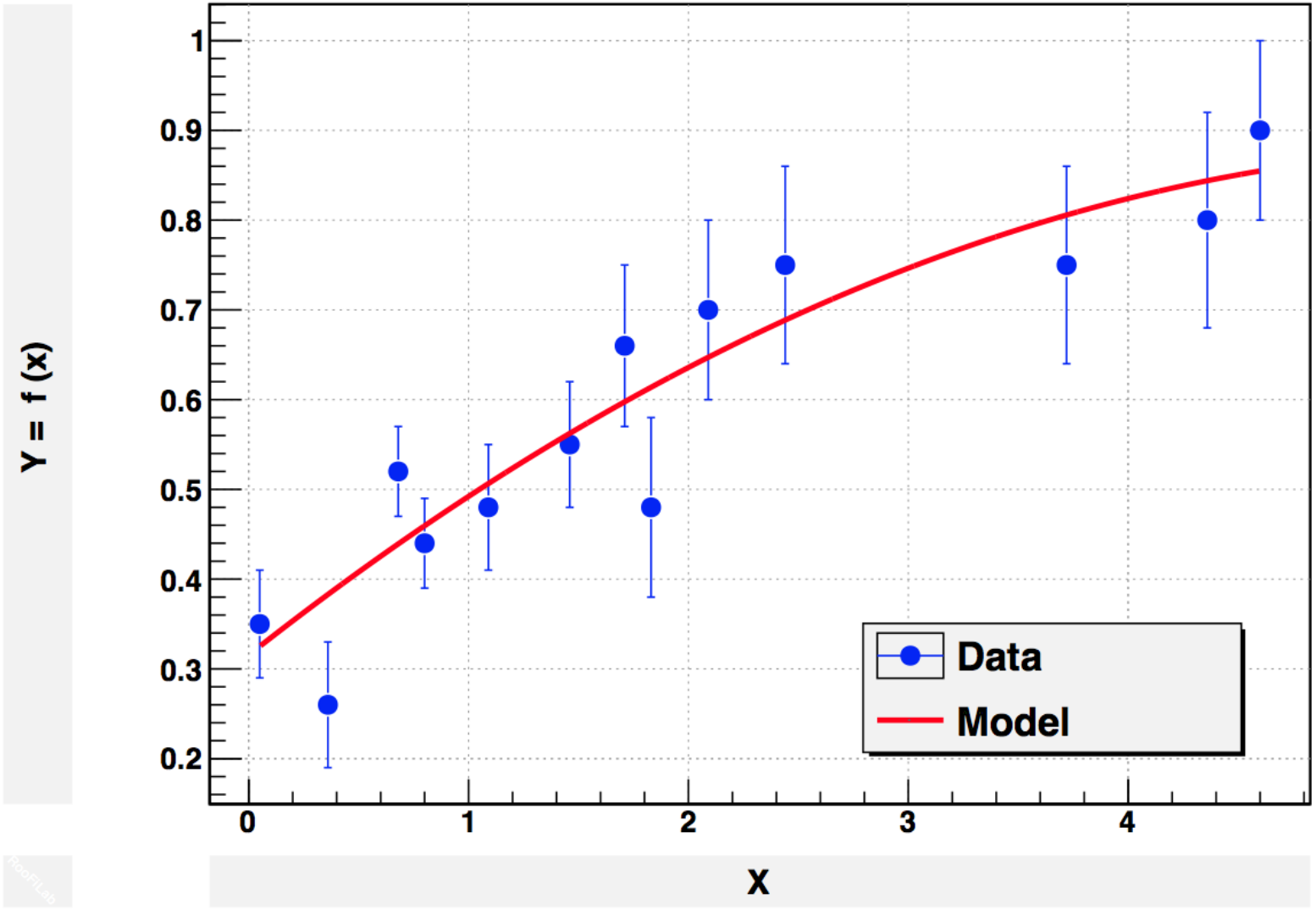
Метод применяется также при аппроксимации функций.



Δy_i – отклонение *i*-ой наблюдаемой точки от модельной кривой

Модель с линейной функцией:

$y = f(x) = a*x + b$


Y = f (x)
X
Data
Model
0
1
2
3
4
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Работа внутри программной оболочки Root с помощью интерпретатора CINT

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

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
```

test1.cpp

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

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;
}
```

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
#ifndef __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");
}
```