ROOT介绍

王思广
Email: siguang@pku.edu.cn

北京大学物理学院

International Summary School on TeV Experimental Physics, Aug. 20, 2014

报告内容

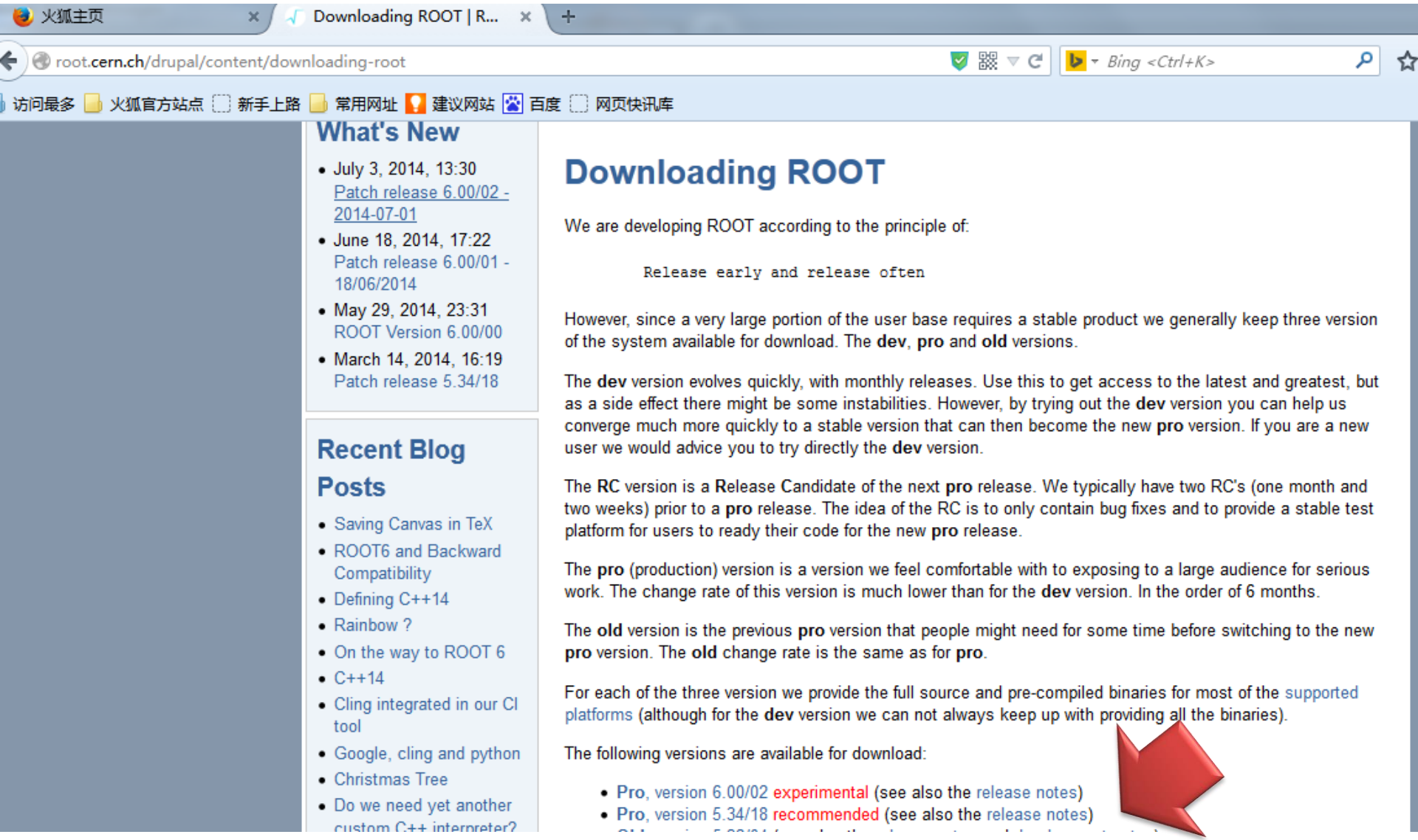
- ROOT自由软件网页
- 利用RooFit进行重叠峰拟合
- ROOT 部分功能展示
- ROOT的安装
- 安装后的运行
- 结束语

ROOT自由软件网页

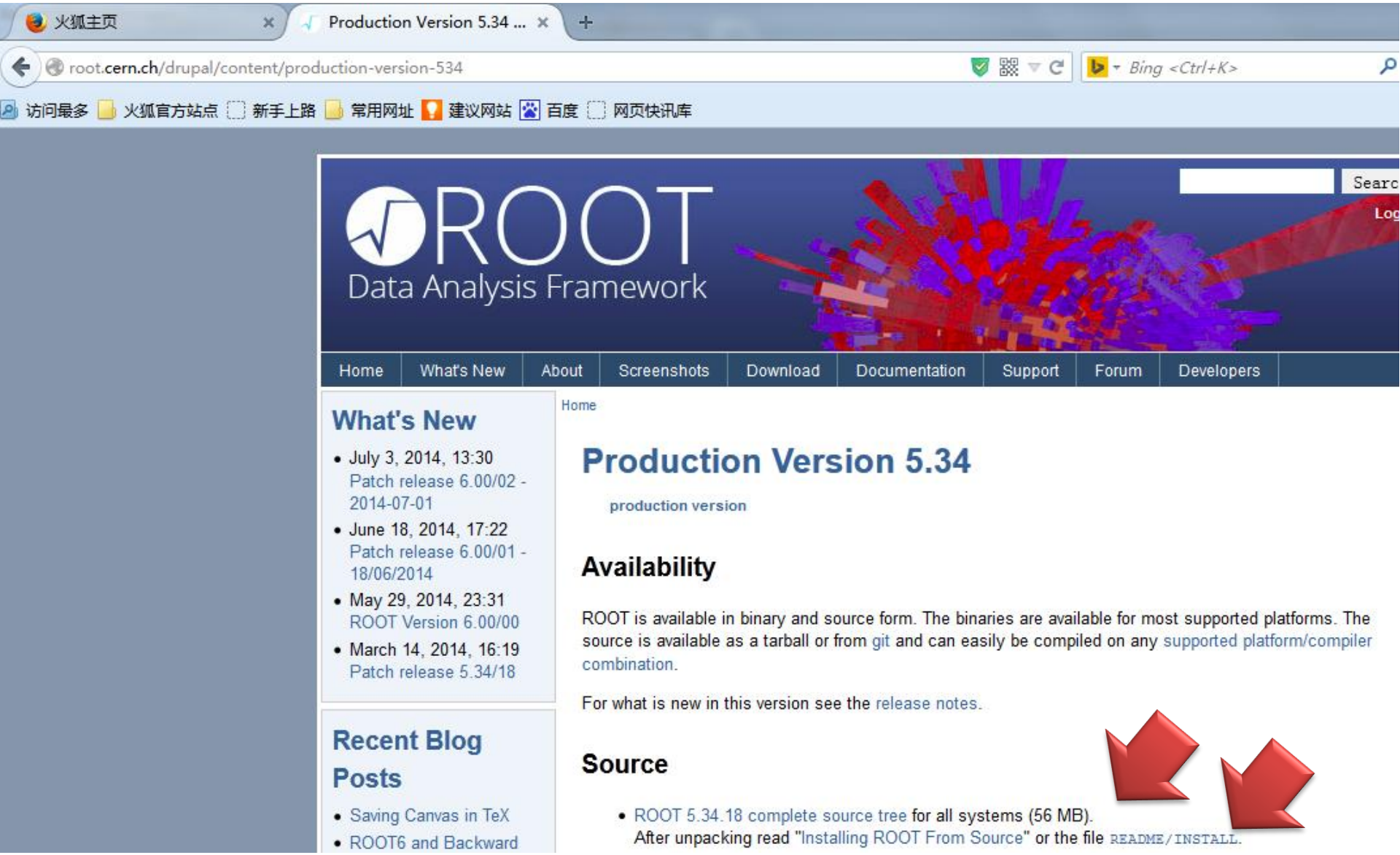
http://root.cern.ch/drupal/

Siguang@pku.edu.cn

ROOT自由软件网页


火狐主页
Downloading ROOT | R...
root.cern.ch/drupal/content/downloading-root
Bing <Ctrl+K>
访问最多
火狐官方站点
新手上路
常用网址
建议网站
百度
网页快讯库
What's New
July 3, 2014, 13:30
Patch release 6.00/02 - 2014-07-01
June 18, 2014, 17:22
Patch release 6.00/01 - 18/06/2014
May 29, 2014, 23:31
ROOT Version 6.00/00
March 14, 2014, 16:19
Patch release 5.34/18
Recent Blog Posts
Saving Canvas in TeX
ROOT6 and Backward Compatibility
Defining C++14
Rainbow ?
On the way to ROOT 6
C++14
Cling integrated in our CI tool
Google, cling and python
Christmas Tree
Do we need yet another
Downloading ROOT
We are developing ROOT according to the principle of:
Release early and release often
However, since a very large portion of the user base requires a stable product we generally keep three version of the system available for download. The dev, pro and old versions.
The dev version evolves quickly, with monthly releases. Use this to get access to the latest and greatest, but as a side effect there might be some instabilities. However, by trying out the dev version you can help us converge much more quickly to a stable version that can then become the new pro version. If you are a new user we would advice you to try directly the dev version.
The RC version is a Release Candidate of the next pro release. We typically have two RC's (one month and two weeks) prior to a pro release. The idea of the RC is to only contain bug fixes and to provide a stable test platform for users to ready their code for the new pro release.
The pro (production) version is a version we feel comfortable with to exposing to a large audience for serious work. The change rate of this version is much lower than for the dev version. In the order of 6 months.
The old version is the previous pro version that people might need for some time before switching to the new pro version. The old change rate is the same as for pro.
For each of the three version we provide the full source and pre-compiled binaries for most of the supported platforms (although for the dev version we can not always keep up with providing all the binaries).
The following versions are available for download:
Pro, version 6.00/02 experimental (see also the release notes)
Pro, version 5.34/18 recommended (see also the release notes)

ROOT自由软件网页


火狐主页
Production Version 5.34 ...
root.cern.ch/drupal/content/production-version-534
Bing <Ctrl+K>
访问最多
火狐官方站点
新手上路
常用网址
建议网站
百度
网页快讯库
ROOT
Data Analysis Framework
Search
Log
Home
What's New
About
Screenshots
Download
Documentation
Support
Forum
Developers
What's New
July 3, 2014, 13:30
Patch release 6.00/02 - 2014-07-01
June 18, 2014, 17:22
Patch release 6.00/01 - 18/06/2014
May 29, 2014, 23:31
ROOT Version 6.00/00
March 14, 2014, 16:19
Patch release 5.34/18
Recent Blog Posts
Saving Canvas in TeX
ROOT6 and Backward
Home
Production Version 5.34
production version
Availability
ROOT is available in binary and source form. The binaries are available for most supported platforms. The source is available as a tarball or from git and can easily be compiled on any supported platform/compiler combination.
For what is new in this version see the release notes.
Source
ROOT 5.34.18 complete source tree for all systems (56 MB).
After unpacking read "Installing ROOT From Source" or the file README/INSTALL.

Project Statistics

http://root.cern.ch/drupal/content/project-statistics

ROOT - Project Cost

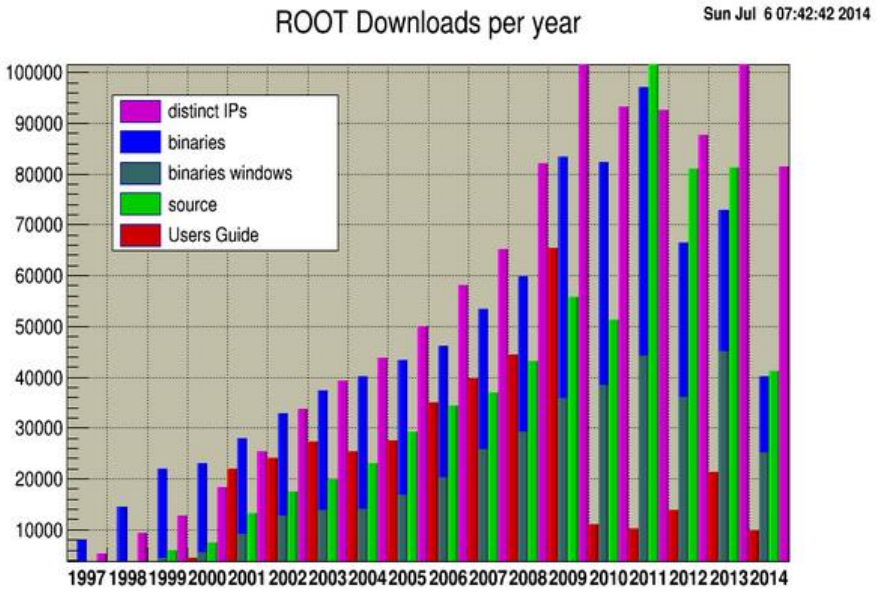
Include	Avg. Salary
Markup And Code	$ 55000 /year
Codebase	**Effort (est.)**
1,744,001 Lines	501 Person Years
Estimated Cost	**$27,578,195**

Updated Jul 05, 2014 more at Ohloh

Download Statistics

download statistics

ROOT Binaries, Source and distinct IPs Statistics

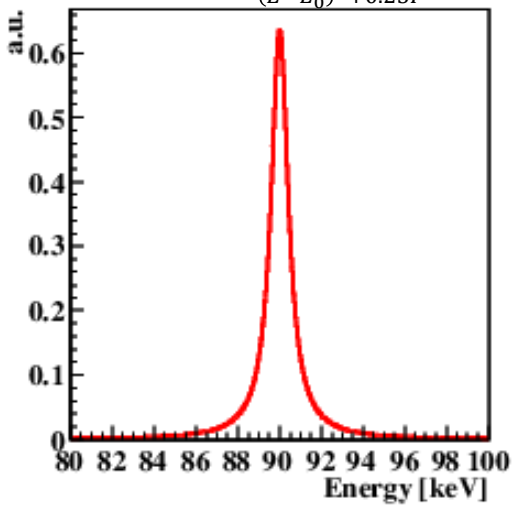


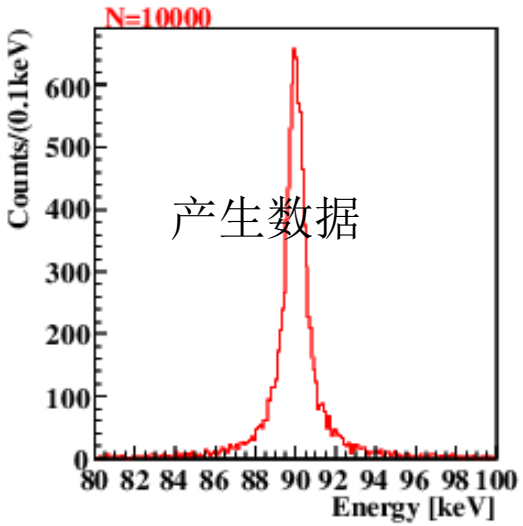
利用RooFit进行重叠峰拟合

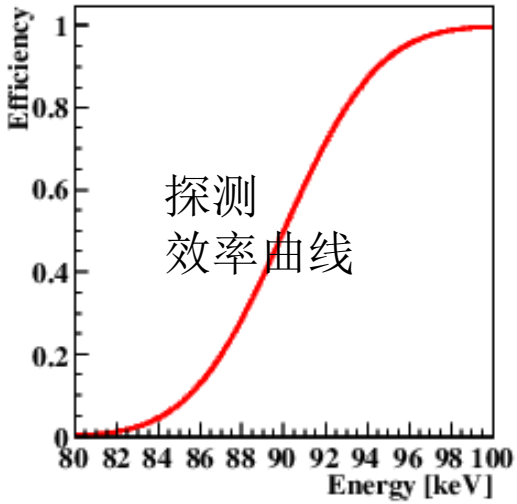
Siguang@pku.edu.cn

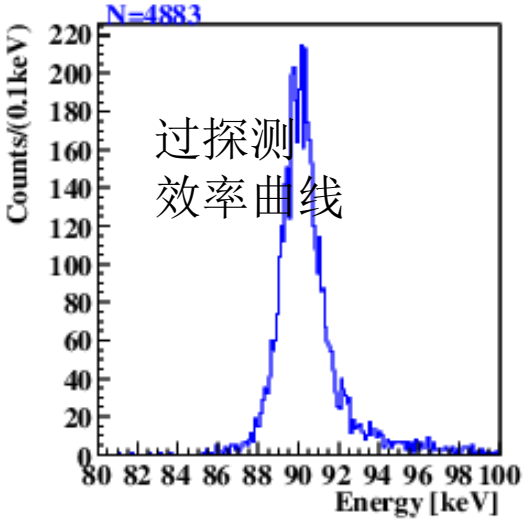
X射线能峰形状

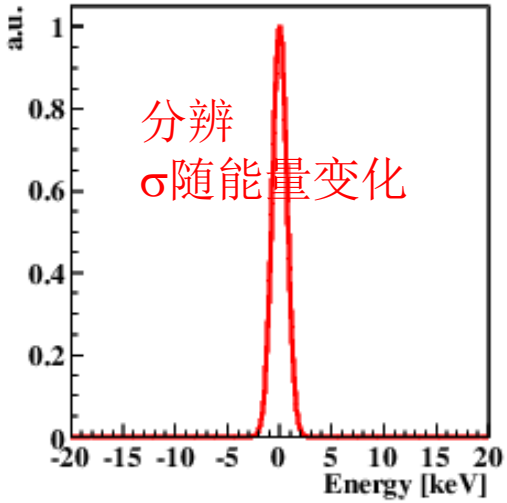
本征形状: $f(E) \propto \frac{1}{(E-E_0)^2+0.25\Gamma^2}$

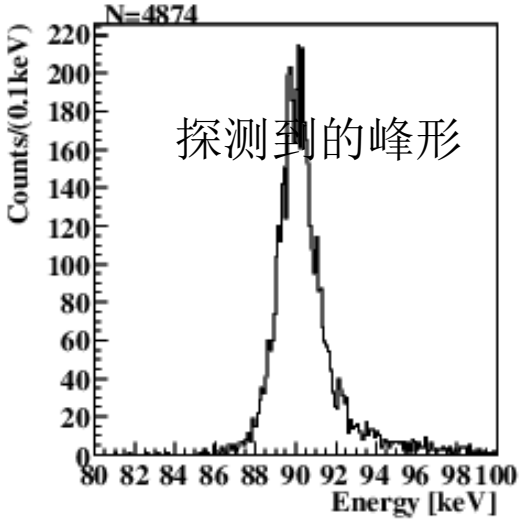






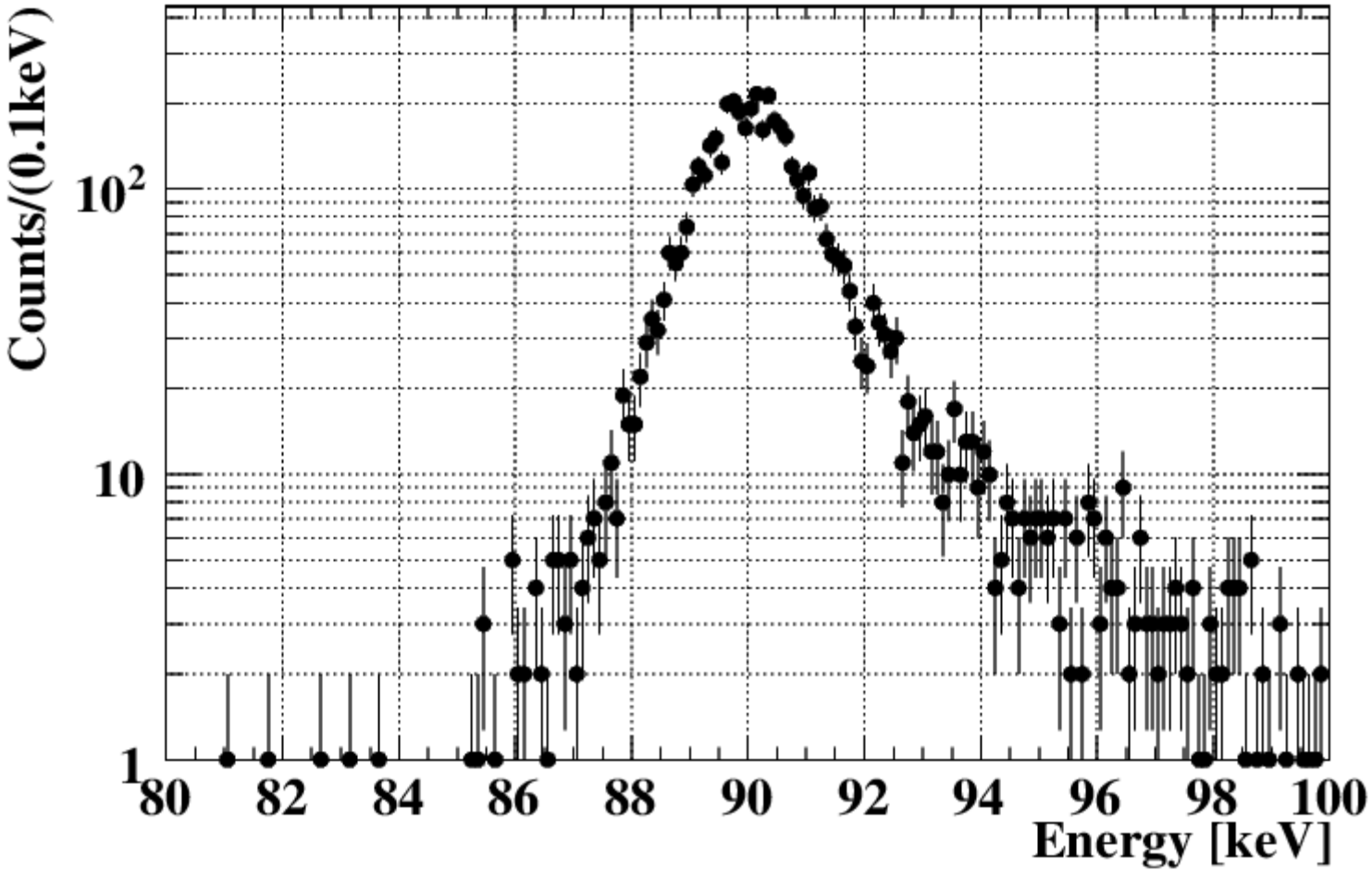






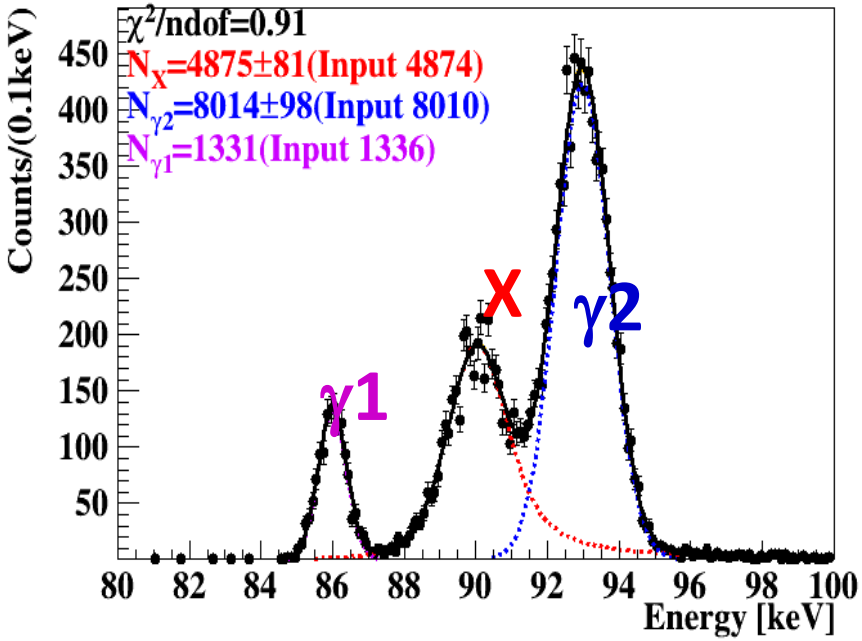
Siguang@pku.edu.cn

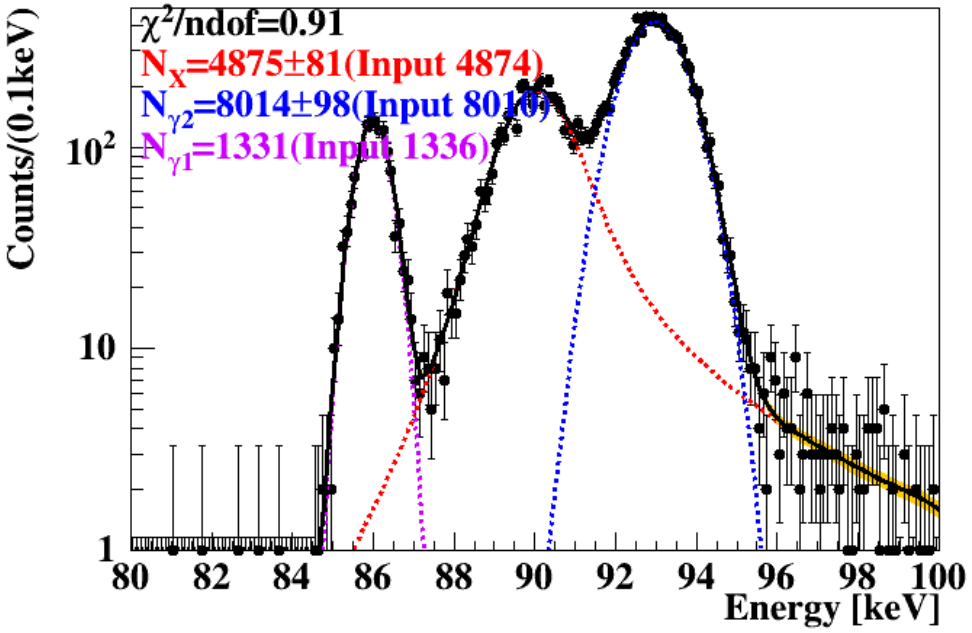
X射线能峰形状(产生)



Siguang@pku.edu.cn

产生并拟合的谱





Siguang@pku.edu.cn

拟合函数

$$N_1 \cdot G(\mathrm{x}, E_1, \sigma(E_1)) \cdot \varepsilon(\mathrm{E}_1) +$$
$$N_2 \cdot \left\{ \left[BW(x, E_2, \Gamma) \cdot \varepsilon(\mathrm{x}) \right] \otimes G(\mathrm{x}, 0, \sigma(E_2)) \right\} +$$
$$N_3 \cdot G(\mathrm{x}, E_3, \sigma(E_3)) \cdot \varepsilon(\mathrm{E}_3)$$

说明：

1） ⊗----卷积功能。利用FFTW （"**Fastest Fourier Transform in the West.**"）

2）分辨是射线能量的函数；对于X射线这里近似用峰位分辨.

3）效率是能量的函数

Siguang@pku.edu.cn

实现代码 （读入能谱）

```
//
//Set plot style----
SetSgStyle();

//
//Read the spectrum.....
TH1D *hSpec = 0;
TH1D *hL = 0;
TH1D *hXray = 0;
TH1D *hR = 0;

Double_t fMeanG1,fMeanG2,fMeanX;
TFile fin("spec.root");
hSpec = (TH1D *)fin.Get("h3Peaks");
hSpec->SetDirectory(0);
hL = (TH1D *)fin.Get("hL");
hL->SetDirectory(0);
hR = (TH1D *)fin.Get("hR");
hR->SetDirectory(0);
hXray = (TH1D *)fin.Get("hXray");
hXray->SetDirectory(0);
fMeanG1 = hL -> GetMean();
fMeanG2 = hR -> GetMean();
fMeanX = hXray -> GetMean();
fin.Close();
Double_t xmin = hSpec->GetXaxis()->GetXmin();
Double_t xmax = hSpec->GetXaxis()->GetXmax();
```

实现代码 （定义X-射线峰）

核心代码

```
RooRealVar x("x","x",xmin,xmax);

//X-ray peak
RooRealVar MeanX("MeanX","Mean of X-ray Peak",fMeanX,fMeanX-5,fMeanX+5);
RooRealVar WidthX("WidthX","Width of X-Ray Peak",1.);//we know the width
RooBreitWigner peakX0("peakX0","X-ray Peak Before Detected",x,MeanX,WidthX);

//Efficiency
RooRealVar a("a","1st coefficency of eff", 90.0,80,100);
RooRealVar b("b","2nd coefficency of eff", 5,1.0,8.0);
RooFormulaVar effFun("effFun","0.5*(TMath::Erf((x-a)/b)+1)",RooArgList(a,b,x)) ;

//Resolution
RooRealVar mg("mg","mg",0) ;
RooRealVar c1("c1","1st coefficency of sg",0.1,0,4);
RooRealVar c2("c2","2nd coefficency of sg",0.05,0,1.);
RooFormulaVar sg("sg",Form("(c1+c2*(MeanX-%g))",xmin),RooArgList(c1,c2,MeanX)) ;
RooGaussian R("R","resolution",x,mg,sg) ;

// Multiply pdf(x) with efficiency in x
RooEffProd peakX0eff("peakX0eff","peakX0 with efficiency",peakX0,effFun) ;

// Construct peakXeff (x) R
// Set #bins to be used for FFT sampling to 10000
x.setBins(10000,"cache") ;
RooFFTConvPdf peakX("peakX","(peakX0*Eff) (X) gauss",x,peakX0eff,R) ;
```


Siguang@pku.edu.cn

实现代码（定义γ_1、γ_2；3峰拟合）

核心代码

```
//1st gamma peak
RooRealVar Mean1("Mean1","Mean of 1st gamma Peak",fMeanG1,fMeanG1-5,fMeanG1+5);
RooFormulaVar Sigma1("Sigma1",Form("(c1+c2*(Mean1-%g))",xmin),RooArgList(c1,c2,Mean1)) ;
RooGaussian peak1("peak1","1st Peak)",x,Mean1,Sigma1);

//
//2nd gamma peak
RooRealVar Mean2("Mean2","Mean of 2nd gamma Peak",fMeanG2,fMeanG2-5,fMeanG2+5);
RooFormulaVar Sigma2("Sigma2",Form("(c1+c2*(Mean2-%g))",xmin),RooArgList(c1,c2,Mean2)) ;
RooGaussian peak2("peak2","2nd Peak)",x,Mean2,Sigma2);

Double_t fN1,fNX,fN2;
fN1 = hSpec->GetEntries()*.33;  fNX = fN1;  fN2 = fN1;
RooRealVar NX("NX","Count under 2nd Peak",fNX,0,3*fNX);
RooRealVar N2("N2","Count under 3rd Peak",fN2,0,3*fN2);
//Using N1 and N2 have same branch ratio and are from same isotope: N1=Eff(x1)/Eff(x2)*N2
RooFormulaVar N1("N1","(TMath::Erf((Mean1-a)/b)+1)/(TMath::Erf((Mean2-a)/b)+1)*N2",RooArgList(a,b,Mean1,Mean2,N2));

RooAddPdf model("model","model",RooArgList(peak1,peakX,peak2),RooArgList(N1,NX,N2)) ;

RooDataHist dh("dh","dh",x,Import(*hSpec)) ;

RooFitResult *frlt = model.fitTo(dh,Save());
```

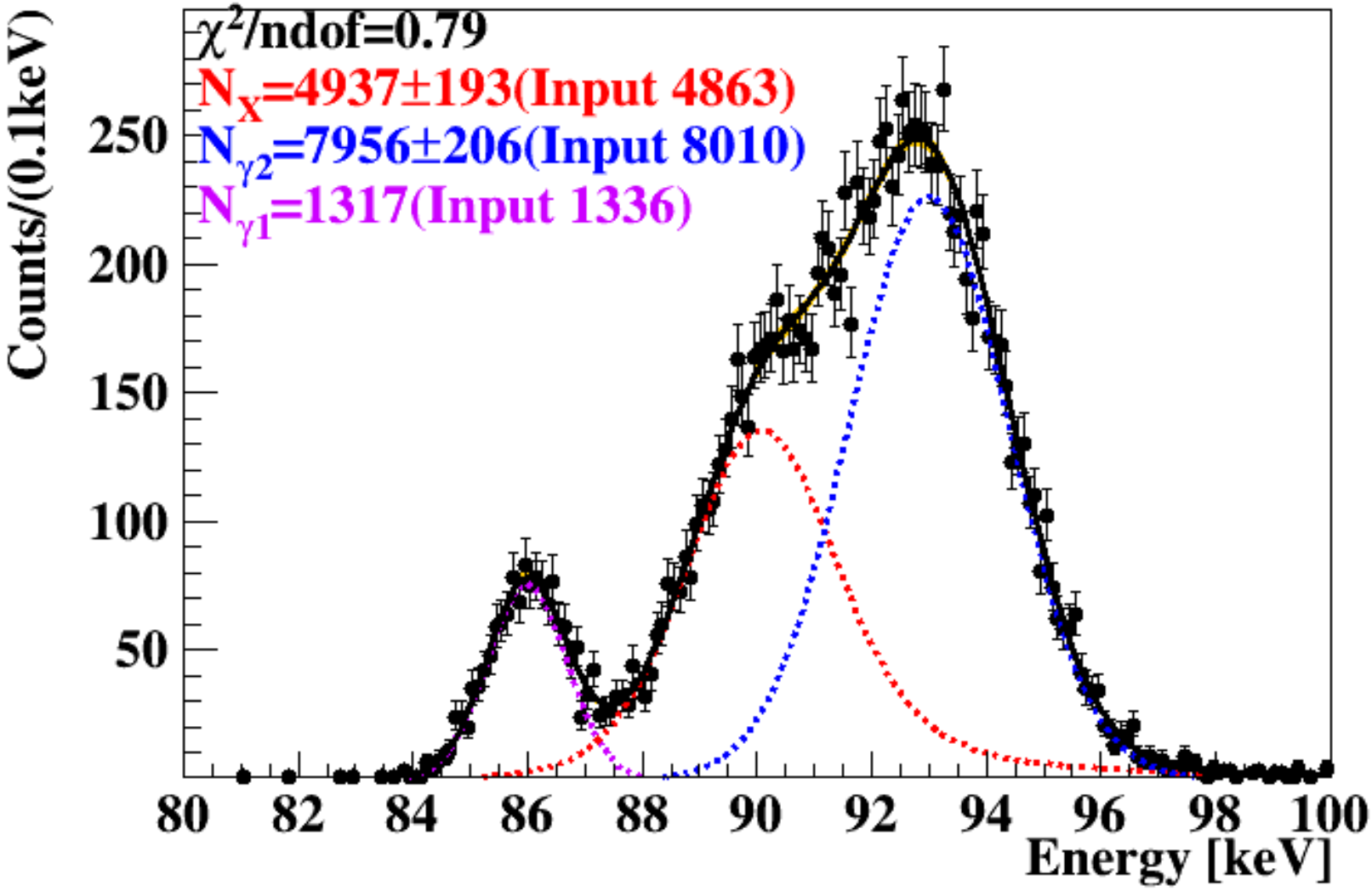
也可以用多CPU同时拟合：

RooFitResult *frlt = model.fitTo(dh,Save(),**NumCPU(2)**);

实现代码（作图）

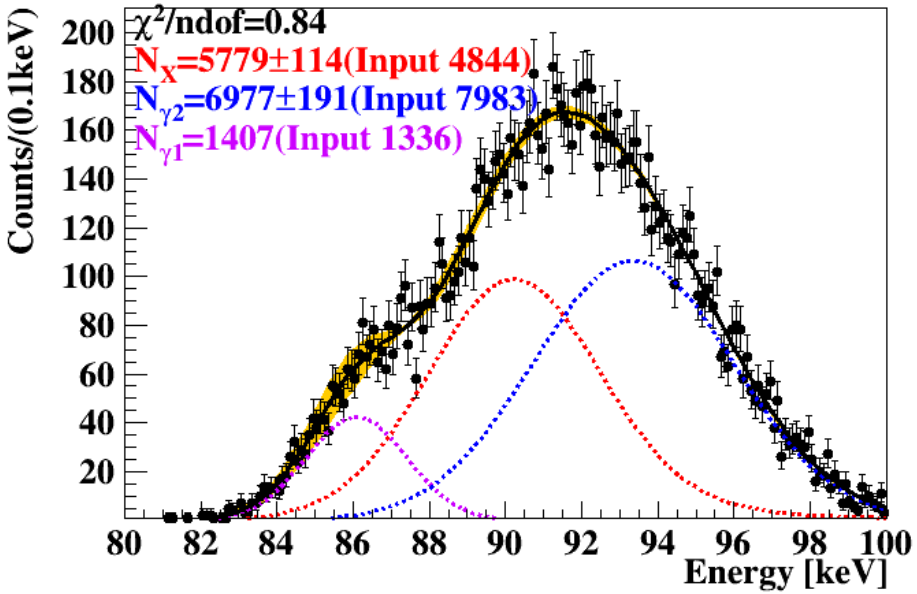
```
RooPlot* frame = x.frame() ;
dh.plotOn(frame) ;
model.plotOn(frame,VisualizeError(*frlt,1),FillColor(kOrange));
dh.plotOn(frame) ;
model.plotOn(frame, Components(peak1),LineStyle(kDashed),LineColor(kViolet));
model.plotOn(frame, Components(peakX),LineStyle(kDashed),LineColor(kRed));
model.plotOn(frame, Components(peak2),LineStyle(kDashed),LineColor(kBlue));
model.plotOn(frame,LineStyle(kSolid),LineColor(kBlack));

TCanvas *cPlot = new TCanvas("cPlot","");
frame->SetMinimum(1);
frame->Draw();

Int_t nParsToFit = (frlt->floatParsFinal()).getSize();
Double_t chi2_red = frame->chiSquare(nParsToFit);//reduced chi-squared = chi2/ndof
txt(0.16,0.9,Form("#chi^{2}/ndof=%.2f",chi2_red));
txt(0.16,0.84,Form("N_{X}=%.0f#pm%.0f(Input %.0f)",NX.getVal(),NX.getError(),hXray->GetEntries()),kRed);
txt(0.16,0.78,Form("N_{#gamma2}=%.0f#pm%.0f(Input %.0f)",N2.getVal(),N2.getError(),hR->GetEntries()),kBlue);
txt(0.16,0.72,Form("N_{#gamma1}=%.0f(Input %.0f)",N1.getVal(),hL->GetEntries()),kViolet);

frame->SetTitle(Form(";%s;%s",hSpec->GetXaxis()->GetTitle(),hSpec->GetYaxis()->GetTitle()));
frlt->Print();
cPlot->Modified();
cPlot->Update();
cPlot->cd();
cPlot->SaveAs("cPlot.pdf");
```

较差分辨下能谱的拟合

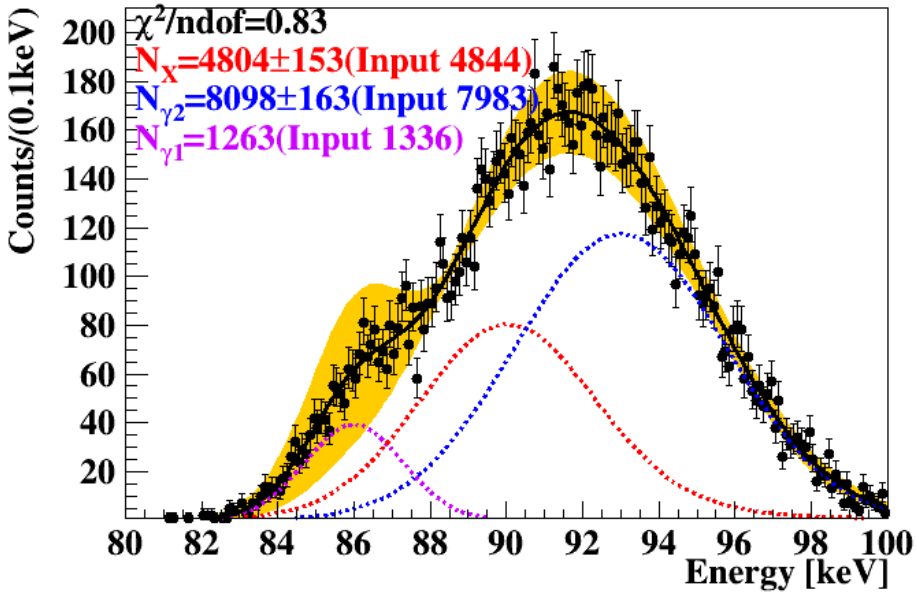


Siguang@pku.edu.cn

更差分辨下能谱的拟合



自由浮动峰位



峰位固定

Siguang@pku.edu.cn

Nuclear Instruments and Methods in Physics Research A 600 (2009) 445–452

Contents lists available at ScienceDirect

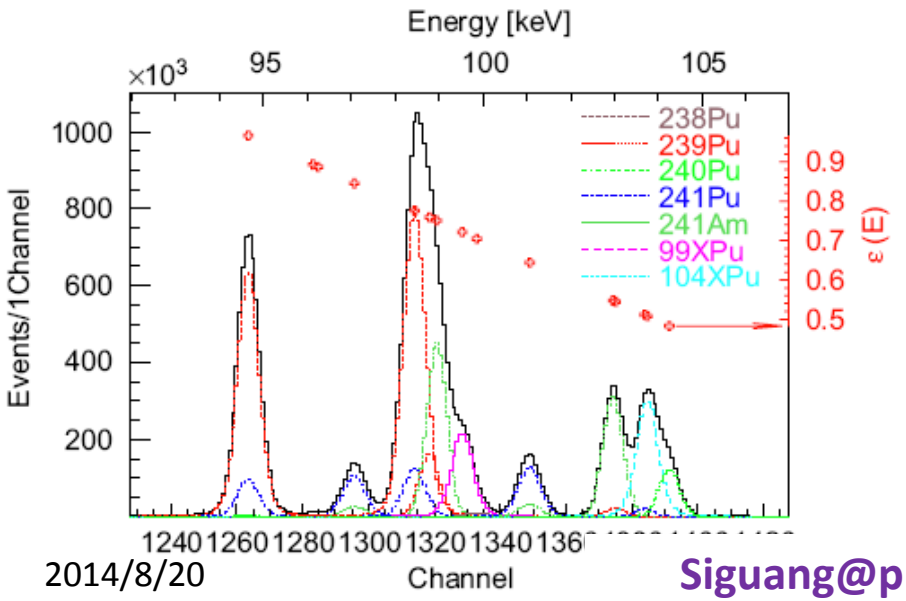
Nuclear Instruments and Methods in Physics Research A

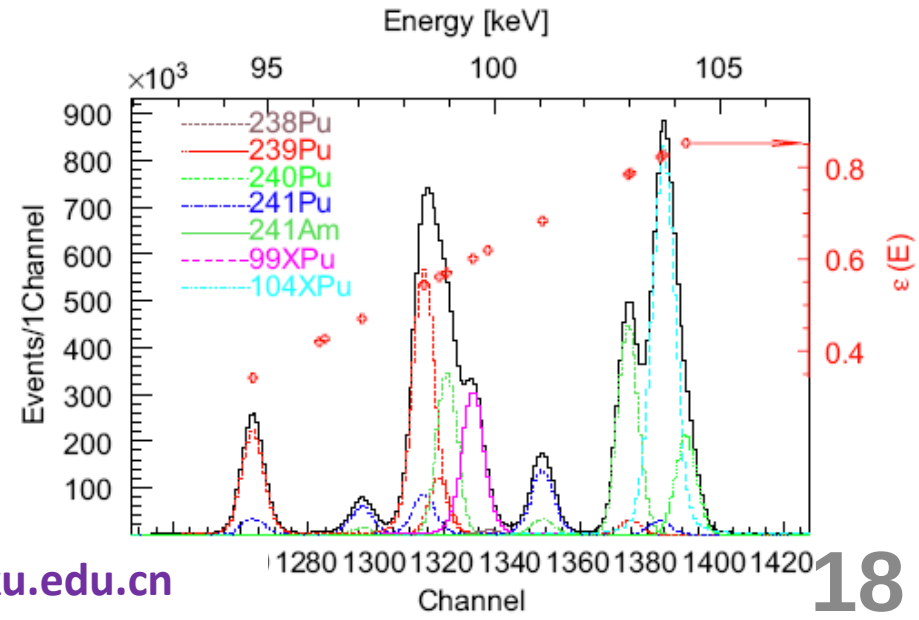
journal homepage: www.elsevier.com/locate/nima

Method for unfolding multiplet regions of X- and γ-ray spectra with a detection efficiency constraint avoiding inflecting the peak shapes for correction results ☆

Si-guang Wang [a,*], Ya-jun Mao [a], Pei-jia Tang [b]

[a] School of Physics and State Key Laboratory of Nuclear Physics and Technology, Peking University, Beijing 100871, PR China
[b] Department of Chemistry, China Institute of Atomic Energy, Beijing 102413, PR China

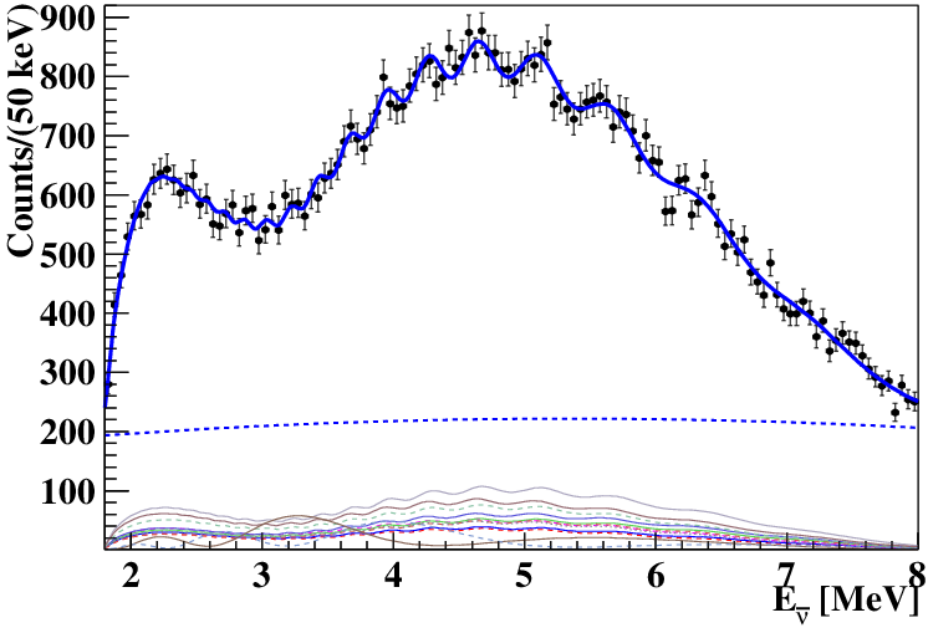


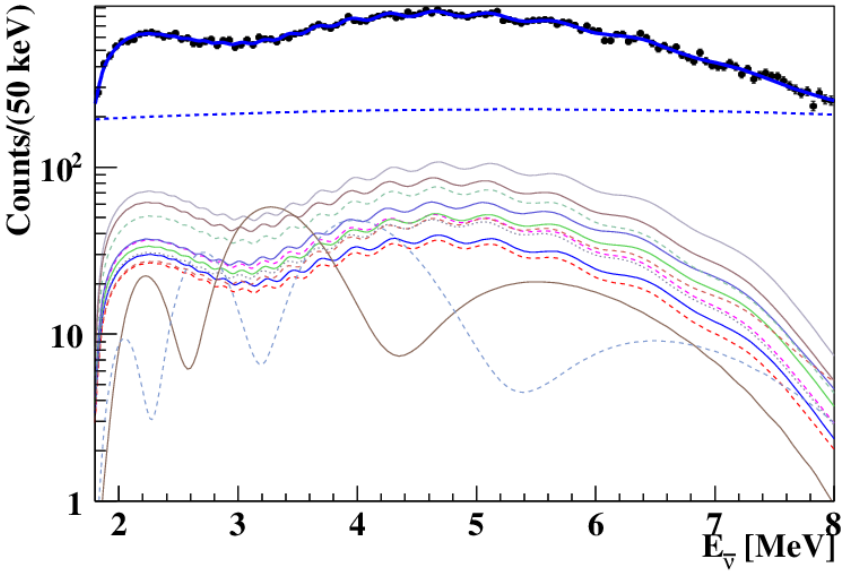


Siguang@pku.edu.cn

分辨随能量变化拟合可以实现

分辨随能量的变化: $R = \frac{3\%}{\sqrt{E_{vis}}}$



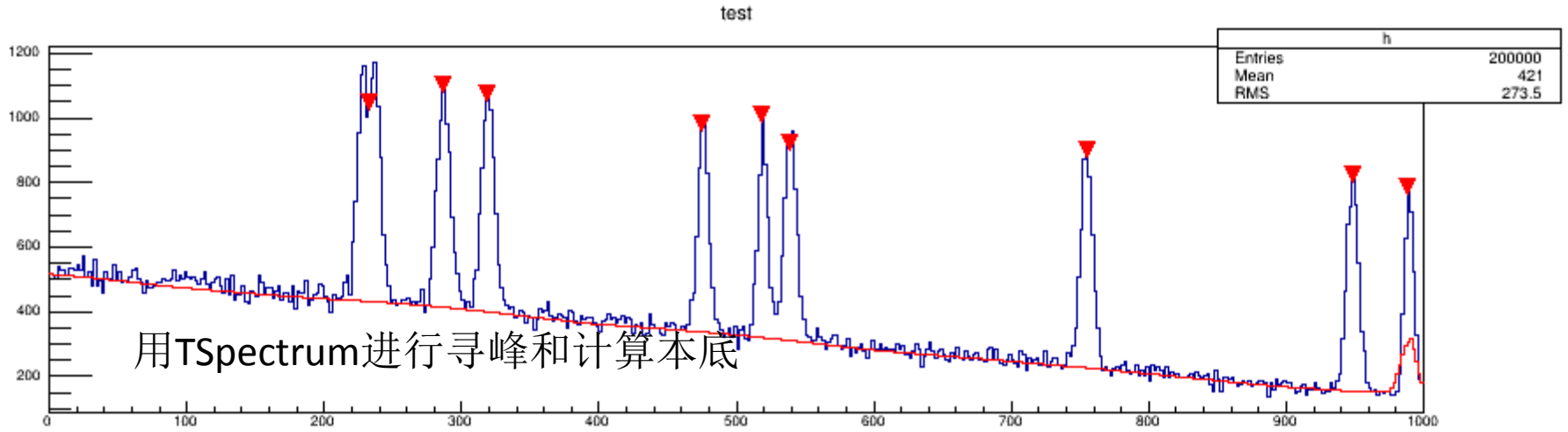


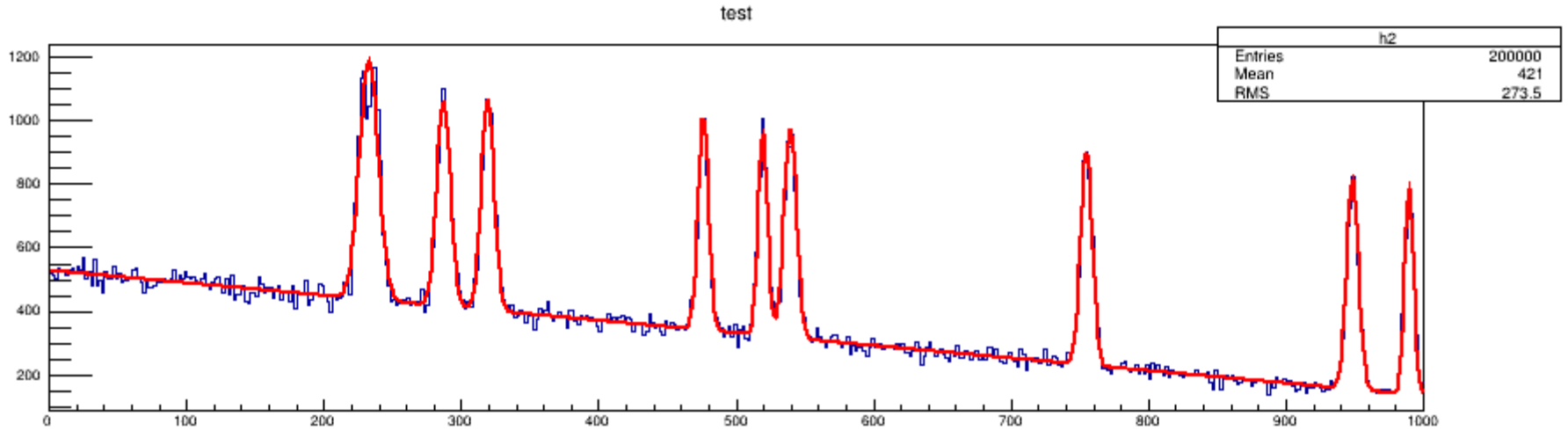
55个自由参数的拟合;

Siguang@pku.edu.cn

ROOT 部分功能展示

ROOT自带的能谱分析类TSpectrum



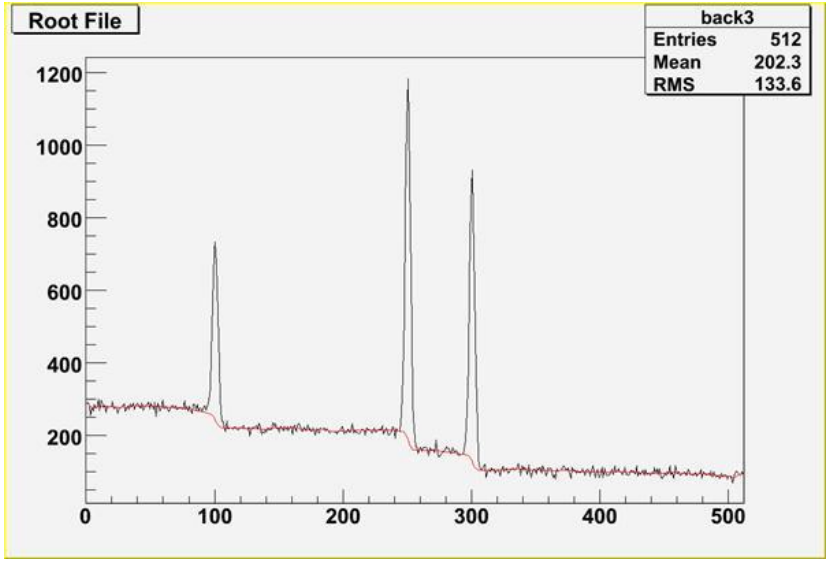


$ROOTSYS/tutorials/spectrum $ root peaks.C

TSpectrum类基本功能之本底估算

const char * Background(float* spectrum, Int_t ssize, Int_t numberIterations, Int_t direction, Int_t filterOrder, bool smoothing, Int_t smoothWindow, bool compton)

Parameters:
spectrum: pointer to the vector of source spectrum
ssize: length of the spectrum vector
numberIterations: maximal width of clipping window,
direction: direction of change of clipping window. Possible values: kBackIncreasingWindow, kBackDecreasingWindow
filterOrder: order of clipping filter. Possible values: kBackOrder2, kBackOrder4, kBackOrder6, kBackOrder8
smoothing: logical variable whether the smoothing operation in the estimation of background will be included. Possible values: kFALSE, kTRUE
smoothWindow: width of smoothing window. Possible values: kBackSmoothing3, kBackSmoothing5, kBackSmoothing7, kBackSmoothing9, kBackSmoothing11, kBackSmoothing13, kBackSmoothing15.
compton: logical variable whether the estimation of Compton edge will be included. Possible values: kFALSE, kTRUE.



http://root.cern.ch/root/html534/TSpectrum.html#TSpectrum:Background

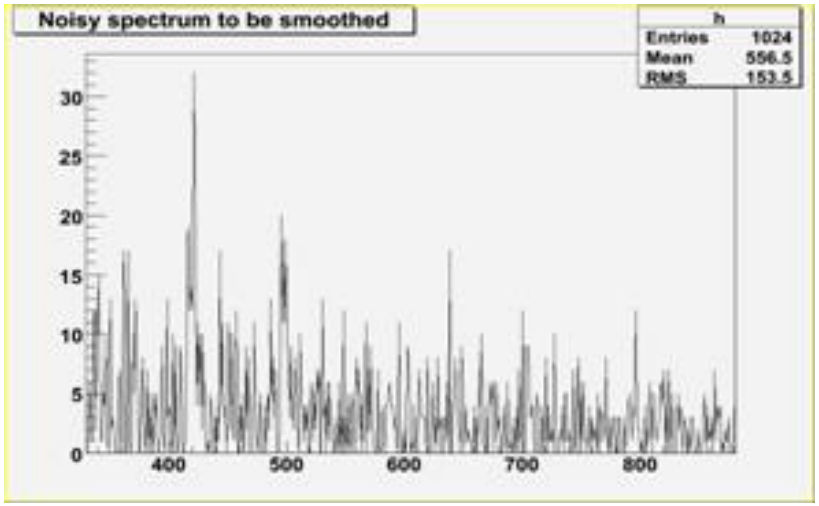
本底估算参考代码

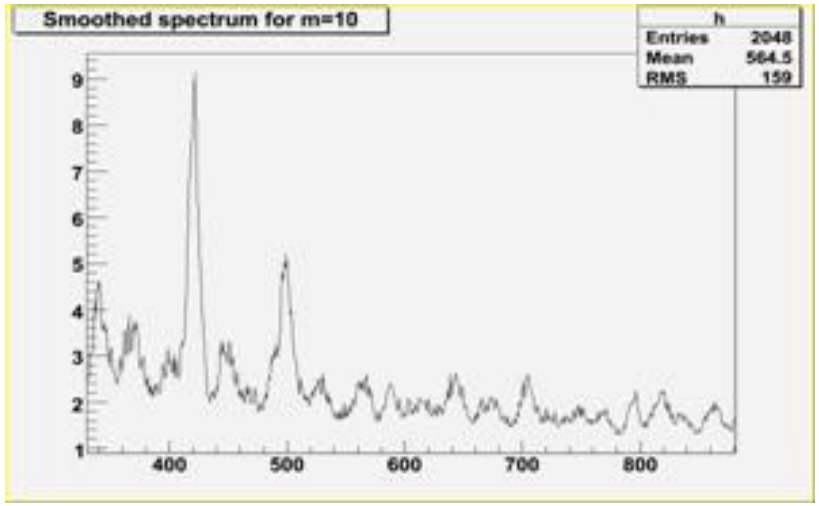
```
// Example to illustrate the background estimator (class TSpectrum) including
// Compton edges. To execute this example, do:
// root > .x Background_compton.C

void Background_compton() {
   Int_t i;
   Double_t nbins = 512;
   Double_t xmin  = 0;
   Double_t xmax  = (Double_t)nbins;
   Float_t * source = new float[nbins];
   TH1F *h = new TH1F("h","",nbins,xmin,xmax);
   TH1F *d1 = new TH1F("d1","",nbins,xmin,xmax);
   TFile *f = new TFile("spectra\\TSpectrum.root");
   h=(TH1F*) f->Get("back3;1");
   TCanvas *background = gROOT->GetListOfCanvases()->FindObject("background");
   if (!background) background = new TCanvas("background",
   "Estimation of background with Compton edges under peaks",10,10,1000,700);
   h->Draw("L");
   TSpectrum *s = new TSpectrum();
   for (i = 0; i < nbins; i++) source[i]=h->GetBinContent(i + 1);
   s->Background(source,nbins,10,kBackDecreasingWindow,kBackOrder8,kTRUE,
   kBackSmoothing5,kTRUE);
   for (i = 0; i < nbins; i++) d1->SetBinContent(i + 1,source[i]);
   d1->SetLineColor(kRed);
   d1->Draw("SAME L");
}
```

TSpectrum类基本功能之能谱光滑

const char* SmoothMarkov(float* source, Int_t ssize, Int_t averWindow)

Parameters:
source: pointer to the array of source spectrum
ssize: length of source array
averWindow: width of averaging smoothing window





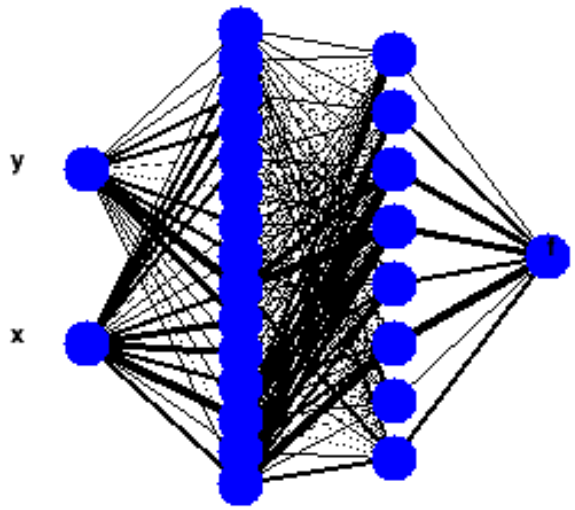
Siguang@pku.edu.cn

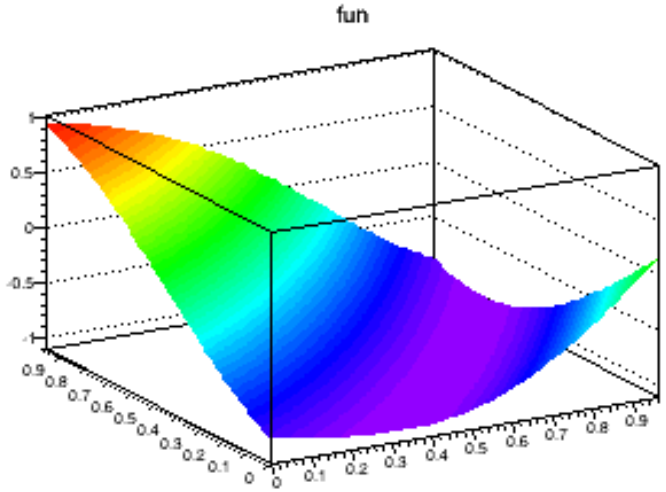
能谱光滑参考代码

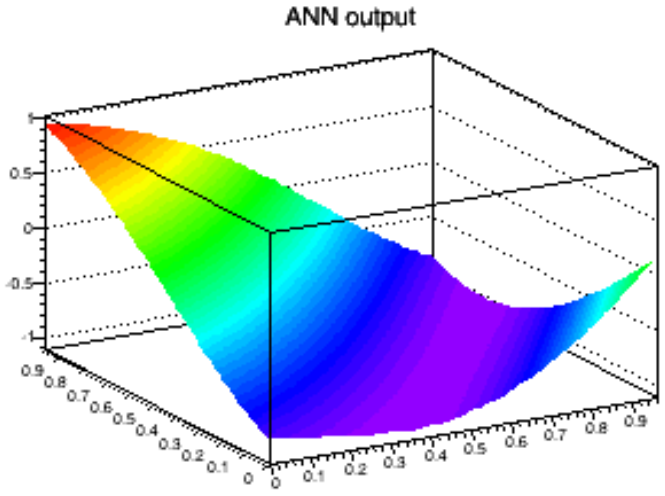
```
// Example to illustrate smoothing using Markov algorithm (class TSpectrum).
// To execute this example, do
// root > .x Smoothing.C
void Smoothing() {
   Int_t i;
   Double_t nbins = 1024;
   Double_t xmin  = 0;
   Double_t xmax  = (Double_t)nbins;
   Float_t * source = new float[nbins];
   TH1F *h = new TH1F("h","Smoothed spectrum for m=3",nbins,xmin,xmax);
   TFile *f = new TFile("spectra\\TSpectrum.root");
   h=(TH1F*) f->Get("smooth1;1");
   for (i = 0; i < nbins; i++) source[i]=h->GetBinContent(i + 1);
   TCanvas *Smooth1 = gROOT->GetListOfCanvases()->FindObject("Smooth1");
   if (!Smooth1) Smooth1 = new TCanvas("Smooth1","Smooth1",10,10,1000,700);
   TSpectrum *s = new TSpectrum();
   s->SmoothMarkov(source,1024,3);  //3, 7, 10
   for (i = 0; i < nbins; i++) h->SetBinContent(i + 1,source[i]);
   h->SetAxisRange(330,880);
   h->Draw("L");
}
```

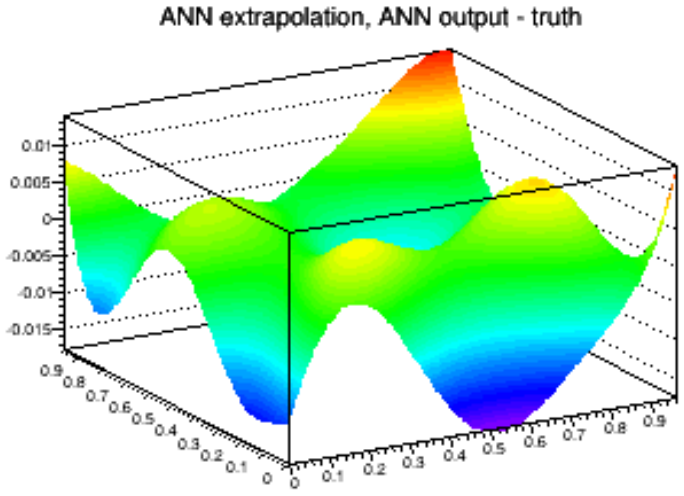
神经网络

利用TMultiLayerPerceptron内插曲面









Siguang@pku.edu.cn

利用TMultiLayerPerceptron内插曲面

```
Double_t theUnknownFunction(Double_t x, Double_t y) {
   return sin((1.7+x)*(x-0.3)-2.3*(y+0.7));
}

void mlpRegression() {
   // create a tree with train and test data.
   // we have two input parameters x and y,
   // and one output value f(x,y)
   TNtuple* t=new TNtuple("tree","tree","x:y:f");
   TRandom r;
   Double_t Wd = 1.0;
   for (Int_t i=0; i<10000; i++) {
      Float_t x=r.Rndm()*Wd;
      Float_t y=r.Rndm()*Wd;

      // fill it with x, y, and f(x,y) - usually this function
      // is not known, and the value of f given an x and a y comes
      // e.g. from measurements
      t->Fill(x,y,theUnknownFunction(x,y));
   }

   // create ANN
   TMultiLayerPerceptron* mlp=new TMultiLayerPerceptron("x,y:15:8:f",t,
      "Entry$%2==1","(Entry$%2)==0");
   mlp->Train(150,"text,graph,update=10");
   mlp->Export("testRlt","C++");
```

剩下的是使用和画图。。。

利用TMultiLayerPerceptron内插曲面

```
// draw difference of ANN's output for (x,y) vs f(x,y) assuming
// the ANN can extrapolate
Int_t N = 30;
const Int_t NN = N*N;
Double_t *vx= new Double_t[NN];
Double_t *vy= new Double_t[NN];
Double_t *delta= new Double_t[NN];
Double_t *vfOrg= new Double_t[NN];
Double_t *vfmlp= new Double_t[NN];
Double_t v[2];
Int_t idx=0;
for (Int_t ix=0; ix<N; ix++) {
   v[0]=Wd*ix/N;
   for (Int_t iy=0; iy<N; iy++) {
      v[1]=Wd*iy/N;
      vx[idx]=v[0];
      vy[idx]=v[1];
      vfOrg[idx] = theUnknownFunction(v[0],v[1]);
      vfmlp[idx] = mlp->Evaluate(0, v);
      delta[idx] = vfmlp[idx]  - vfOrg[idx];
      idx++;
   }
}
```

利用TMultiLayerPerceptron内插曲面

```
   TCanvas *c1 = new TCanvas("c1","");
   c1->Divide(2,2);
   c1->cd(1);
   mlp->Draw();
   c1->cd(2);
   TGraph2D* g2Org=new TGraph2D("g2Org",
                                "fun",
                                idx, vx, vy, vfOrg);
   g2Org->Draw("TRI2");
   c1->cd(3);
   TGraph2D* g2mlp=new TGraph2D("g2mlp",
                                "ANN output",
                                idx, vx, vy, vfmlp);
   g2mlp->Draw("TRI2");
   c1->cd(4);
   TGraph2D* g2Extrapolate=new TGraph2D("ANN extrapolation",
                                "ANN extrapolation, ANN output - truth",
                                idx, vx, vy, delta);
   g2Extrapolate->Draw("TRI2");

   delete[] vx;
   delete[] vy;
   delete[] delta;
   delete[] vfOrg;
   delete[] vfmlp;
}
```

注：在$ROOTSYS/tutorials/mlp$ root mlpRegression.C 的基础上修改

利用TMultiLayerPerceptron内插曲面

可输出C++类作为结果提供使用

testRlt.h

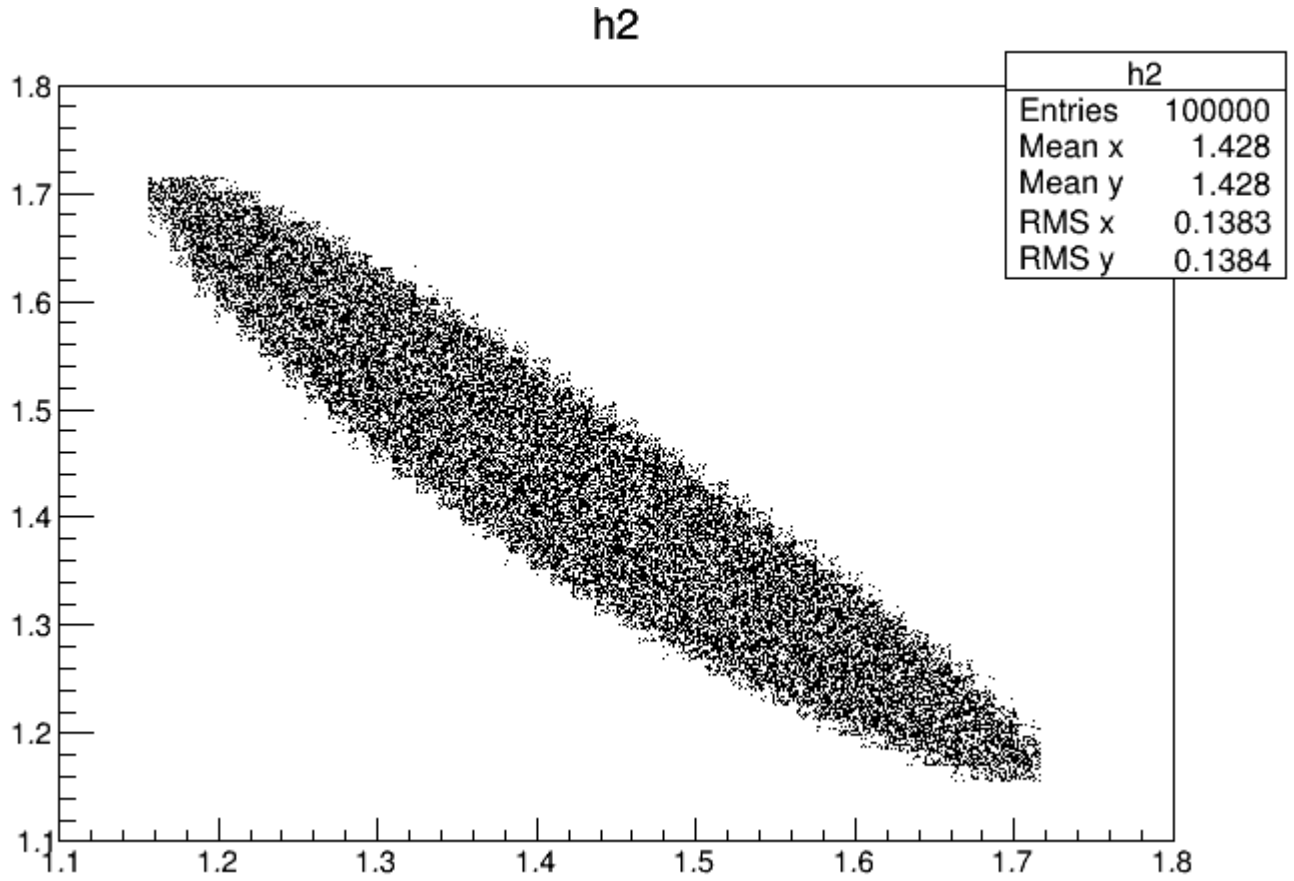
```
class testRlt {
public:
   testRlt() {}
   ~testRlt() {}
   double Value(int index,double in0,double in1);
   double Value(int index, double* input);
private:
   double input0;
   double input1;
   double neuron0x2cd6450();
   double neuron0x2cd6790();
   double input0x2ce9850();
   double neuron0x2ce9850();
   double input0x2ce9b80();
   double neuron0x2ce9b80();
   double input0x2ce9f40();
   double neuron0x2ce9f40();
   double input0x2cea300();
   double neuron0x2cea300();
   double input0x2cea6c0();
   double neuron0x2cea6c0();
   double input0x2ceaa80();
   double neuron0x2ceaa80();
   double input0x2ceae40();
   double neuron0x2ceae40();
   double input0x2ceb200();
   double neuron0x2ceb200();
```

testRlt.cxx

```
double testRlt::Value(int index,double in0,double in1) {
   input0 = (in0 - 0)/1;
   input1 = (in1 - 0)/1;
   switch(index) {
     case 0:
         return neuron0x2cf0fa0();
     default:
         return 0.;
   }
}

double testRlt::Value(int index, double* input) {
   input0 = (input[0] - 0)/1;
   input1 = (input[1] - 0)/1;
   switch(index) {
     case 0:
         return neuron0x2cf0fa0();
     default:
         return 0.;
   }
}

double testRlt::neuron0x2cf0fa0() {
   double input = input0x2cf0fa0();
   return (input * 1)+0;
}

double testRlt::input0x2cf0fa0() {
   double input = -1.04701;
   input += synapse0x2cf12e0();
   input += synapse0x2cf1320();
   input += synapse0x2cf1360();
   input += synapse0x2cf13a0();
   input += synapse0x2cf13e0();
   input += synapse0x2cf1420();
   input += synapse0x2cf1460();
   input += synapse0x2cf14a0();
   return input;
}
```

PhaseSpace

$ROOTSYS/tutorials/physics/PhaseSpace.C

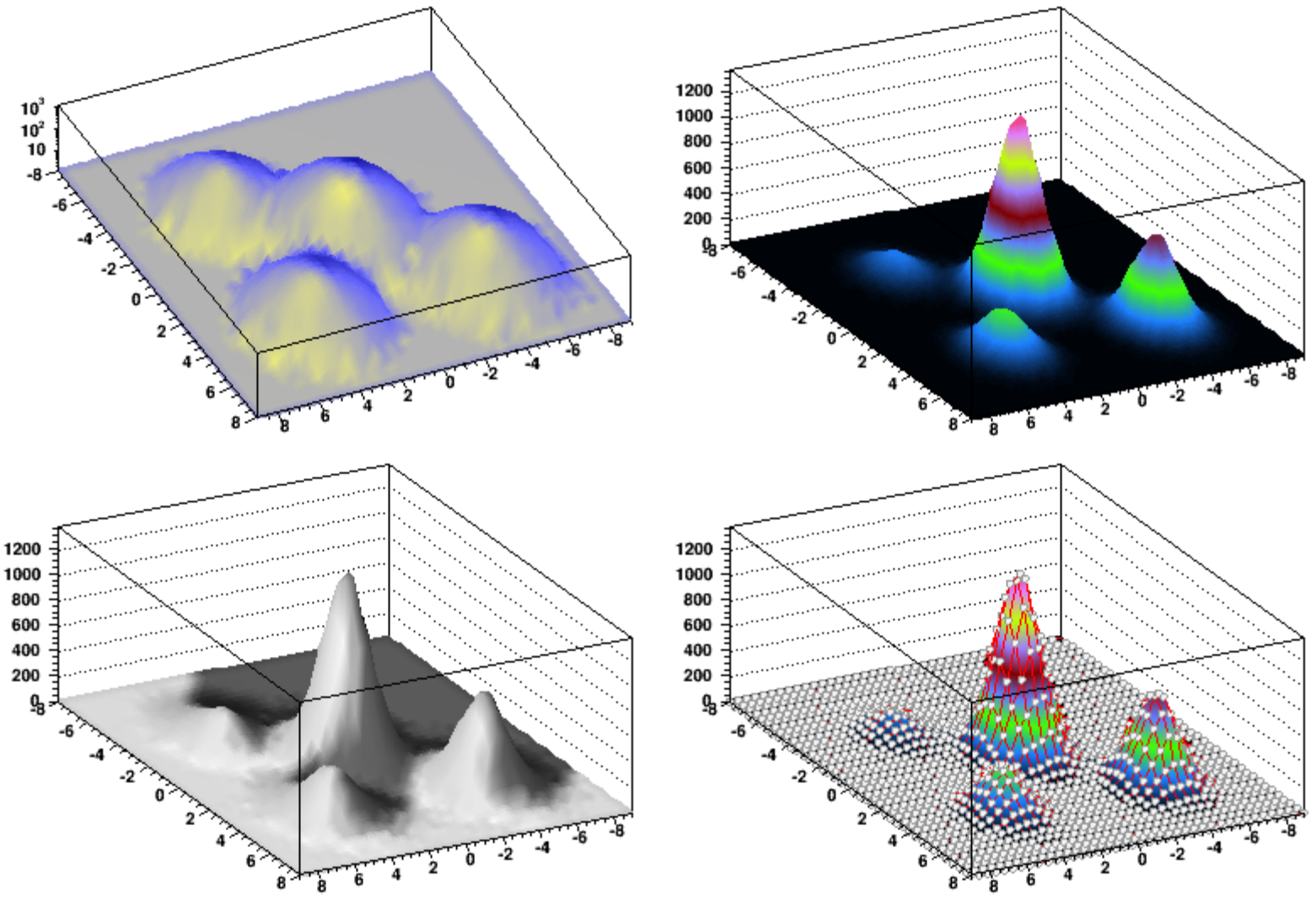


Siguang@pku.edu.cn

$ROOTSYS/tutorials/physics/PhaseSpace.C

```
void PhaseSpace() {
// example of use of TGenPhaseSpace
//Author: Valerio Filippini

   if (!gROOT->GetClass("TGenPhaseSpace")) gSystem.Load("libPhysics");

   TLorentzVector target(0.0, 0.0, 0.0, 0.938);
   TLorentzVector beam(0.0, 0.0, .65, .65);
   TLorentzVector W = beam + target;

   //(Momentum, Energy units are Gev/C, GeV)
   Double_t masses[3] = { 0.938, 0.139, 0.139} ;

   TGenPhaseSpace event;
   event.SetDecay(W, 3, masses);

   TH2F *h2 = new TH2F("h2","h2", 50,1.1,1.8, 50,1.1,1.8);

   for (Int_t n=0;n<100000;n++) {
      Double_t weight = event.Generate();

      TLorentzVector *pProton = event.GetDecay(0);

      TLorentzVector *pPip    = event.GetDecay(1);
      TLorentzVector *pPim    = event.GetDecay(2);

      TLorentzVector pPPip = *pProton + *pPip;
      TLorentzVector pPPim = *pProton + *pPim;

      h2->Fill(pPPip.M2() ,pPPim.M2() ,weight);
   }
   h2->Draw();
}
```

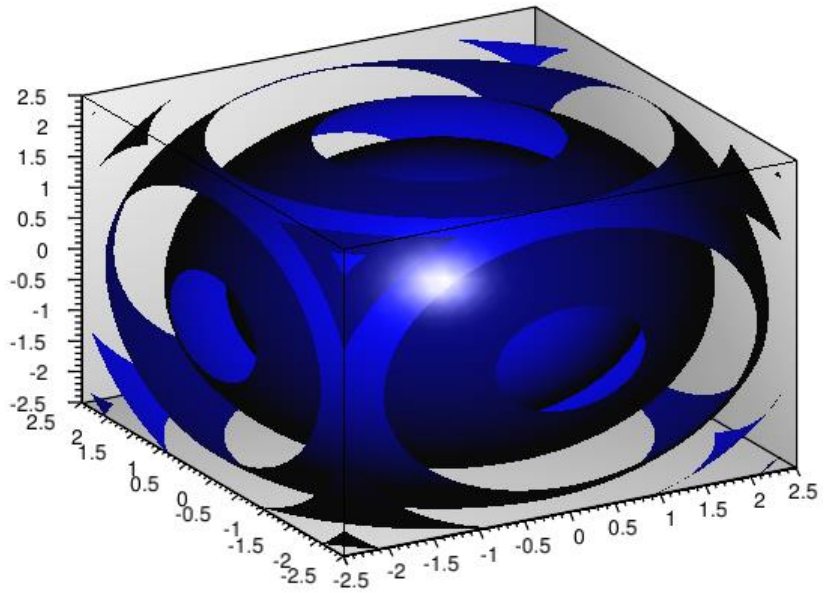
2D 显示



$ROOTSYS/tutorials/spectrum$ root spectrumpainter.C

Siguang@pku.edu.cn

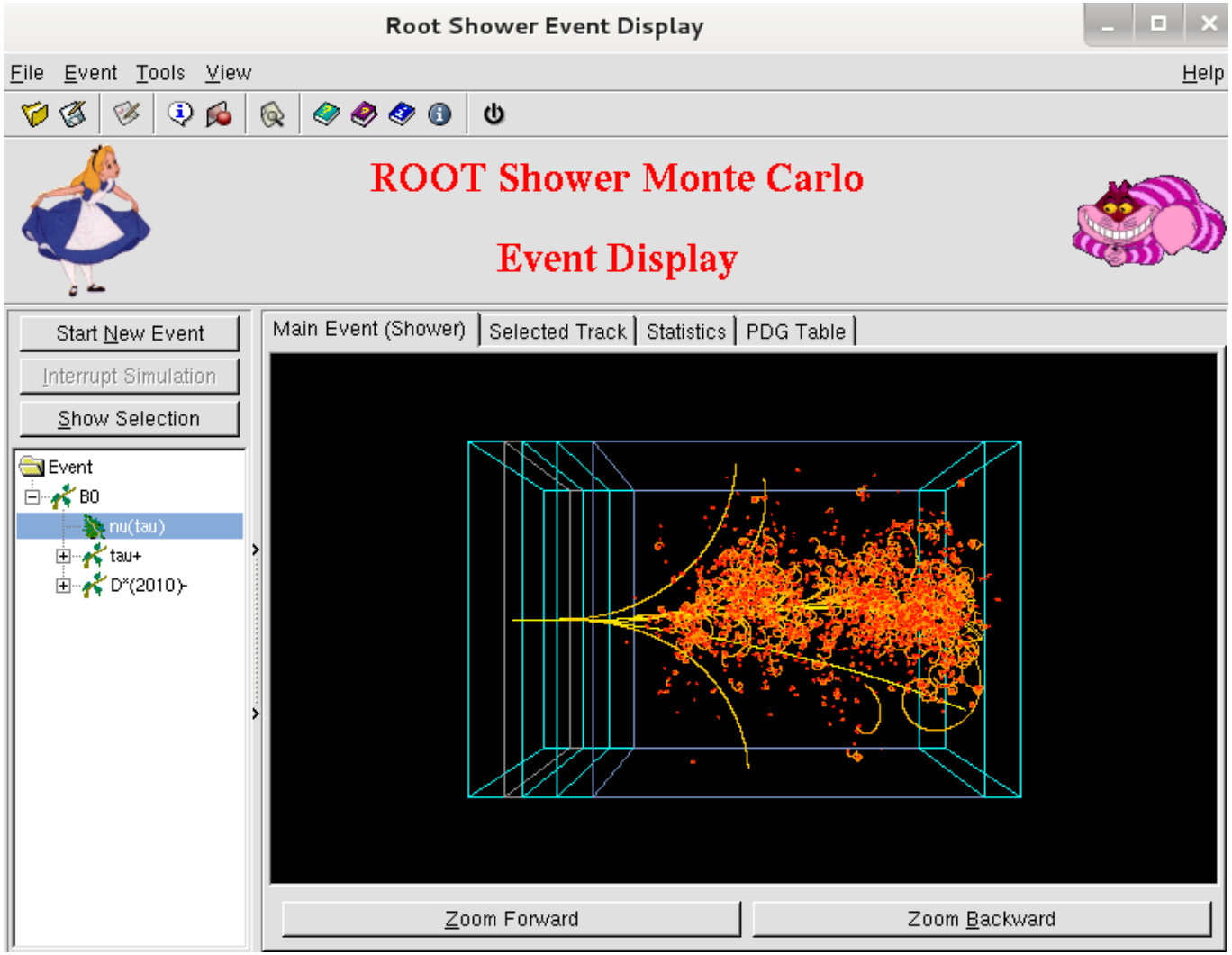
画OpenGL图



ROOT 替我们做了大量工作!

```
void glFun(){
   // after this command all legos surfaces are automatically rendered with OpenGL.
   gStyle->SetCanvasPreferGL(kTRUE);
   TCanvas *c2 = new TCanvas("glc2","");
   TF3 *fun4 = new TF3("fun4","sin(x * x + y * y + z * z - 4)",
                       -2.5, 2.5, -2.5, 2.5, -2.5, 2.5);
   fun4->SetFillColor(kBlue);
   fun4->Draw("gl");
}
```

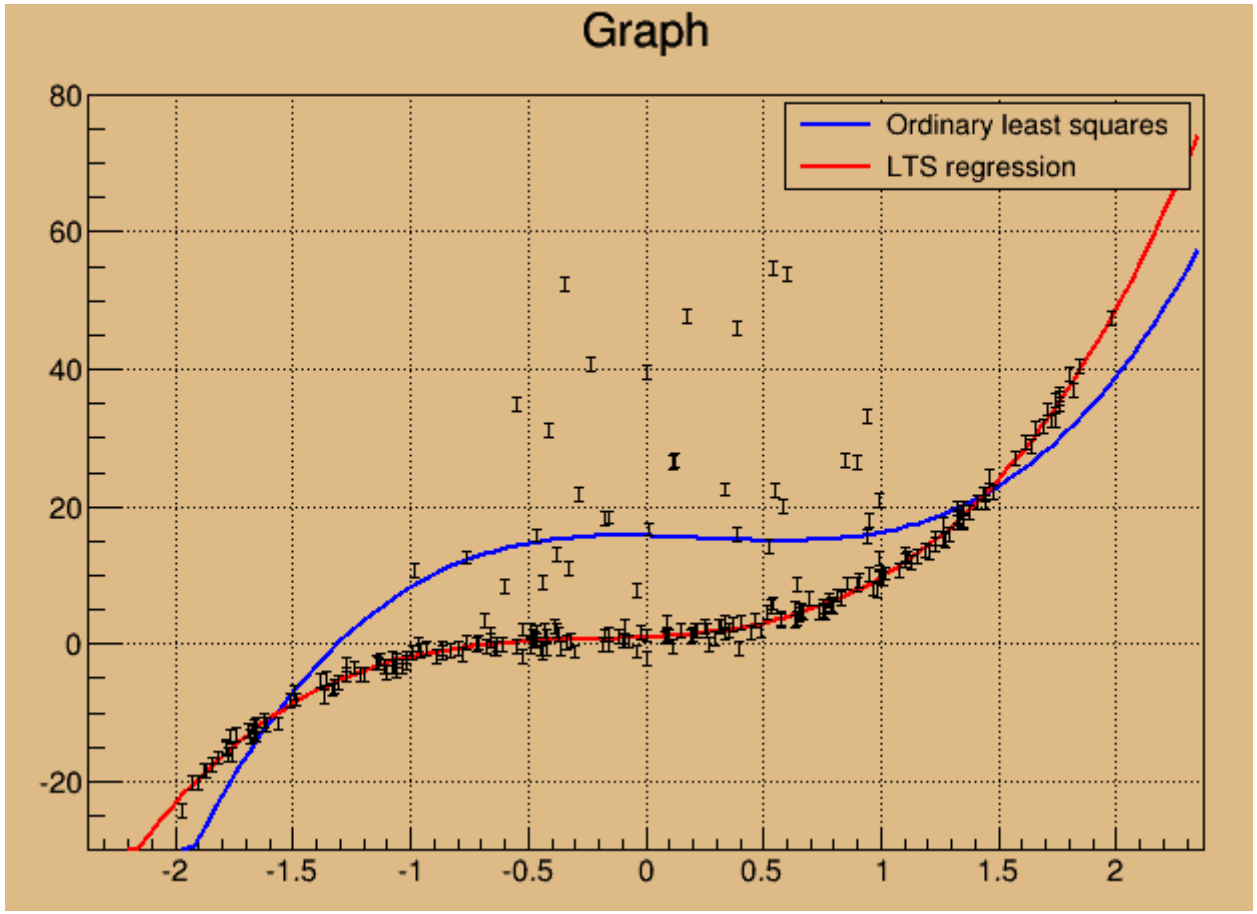
图形界面



$ROOTSYS/test/RootShower$ ./RootShower

Siguang@pku.edu.cn

$ROOTSYS/tutorials/fit/fitLinearRobust.C



Siguang@pku.edu.cn

$ROOTSYS/tutorials/fit/fitLinearRobust.C

数据准备

```
#include "TRandom.h"
#include "TGraphErrors.h"
#include "TF1.h"
#include "TCanvas.h"
#include "TLegend.h"

void fitLinearRobust()
{
//This tutorial shows how the least trimmed squares regression,
//included in the TLinearFitter class, can be used for fitting
//in cases when the data contains outliers.
//Here the fitting is done via the TGraph::Fit function with option "rob":
//If you want to use the linear fitter directly for computing
//the robust fitting coefficients, just use the TLinearFitter::EvalRobust
//function instead of TLinearFitter::Eval
//Author: Anna Kreshuk

   //First generate a dataset, where 20% of points are spoiled by large
   //errors
   Int_t npoints = 250;
   Int_t fraction = Int_t(0.8*npoints);
   Double_t *x = new Double_t[npoints];
   Double_t *y = new Double_t[npoints];
   Double_t *e = new Double_t[npoints];
  TRandom r;
  Int_t i;
  for (i=0; i<fraction; i++){
     //the good part of the sample
     x[i]=r.Uniform(-2, 2);
     e[i]=1;
     y[i]=1 + 2*x[i] + 3*x[i]*x[i] + 4*x[i]*x[i]*x[i] + e[i]*r.Gaus();
  }
  for (i=fraction; i<npoints; i++){
     //the bad part of the sample
     x[i]=r.Uniform(-1, 1);
     e[i]=1;
     y[i] = 1 + 2*x[i] + 3*x[i]*x[i] + 4*x[i]*x[i]*x[i] + r.Landau(10, 5);
  }
```

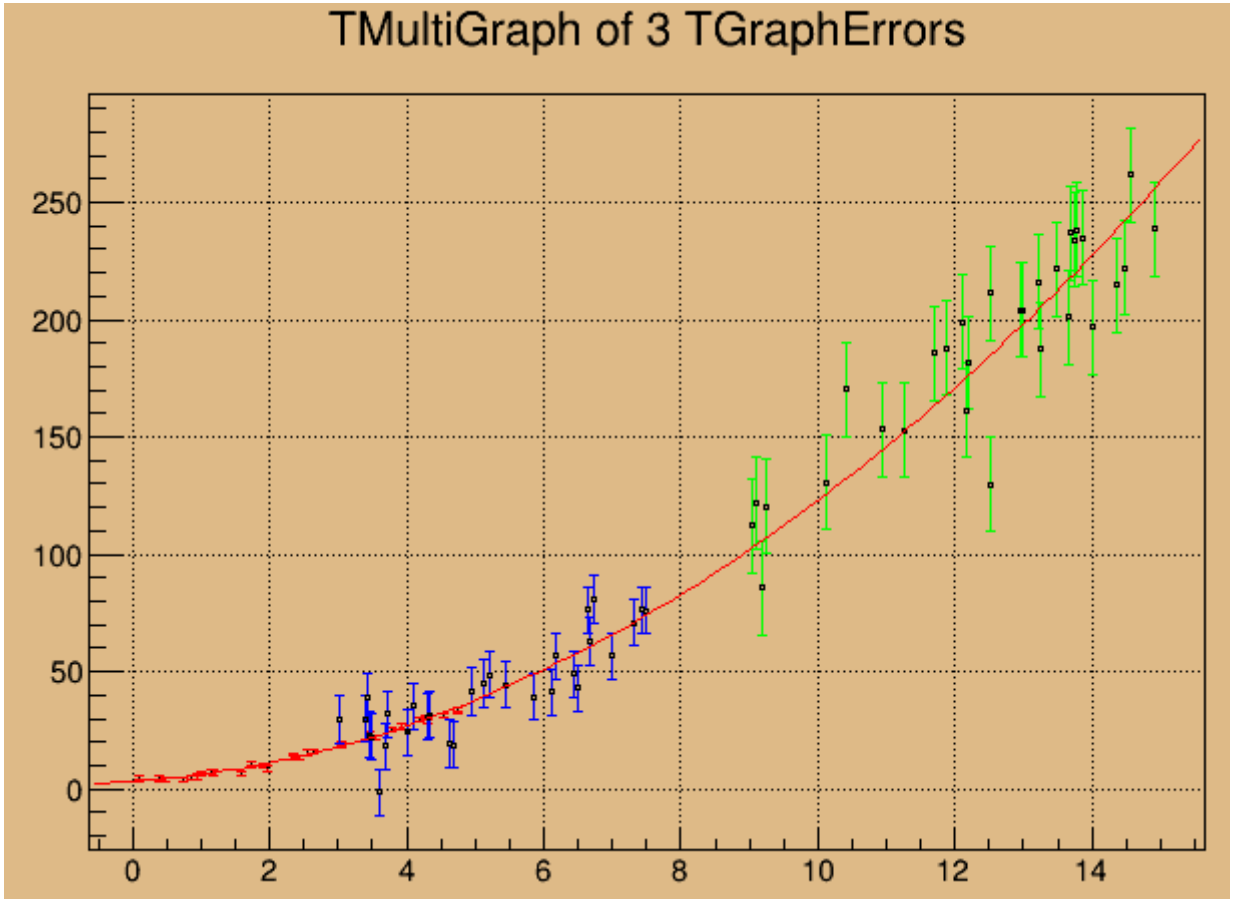
$ROOTSYS/tutorials/fit/fitLinearRobust.C

```
TGraphErrors *grr = new TGraphErrors(npoints, x, y, 0, e);
grr->SetMinimum(-30);
grr->SetMaximum(80);
TF1 *ffit1 = new TF1("ffit1", "pol3", -5, 5);
TF1 *ffit2 = new TF1("ffit2", "pol3", -5, 5);
ffit1->SetLineColor(kBlue);
ffit2->SetLineColor(kRed);
TCanvas *myc = new TCanvas("myc", "Linear and robust linear fitting");
myc->SetFillColor(42);
myc->SetGrid();
grr->Draw("ap");
//first, let's try to see the results of ordinary least-squares fit:
printf("Ordinary least squares:\n");
grr->Fit(ffit1);
//the fitted function doesn't really follow the pattern of the data
//and the coefficients are far from the real ones
```

$ROOTSYS/tutorials/fit/fitLinearRobust.C

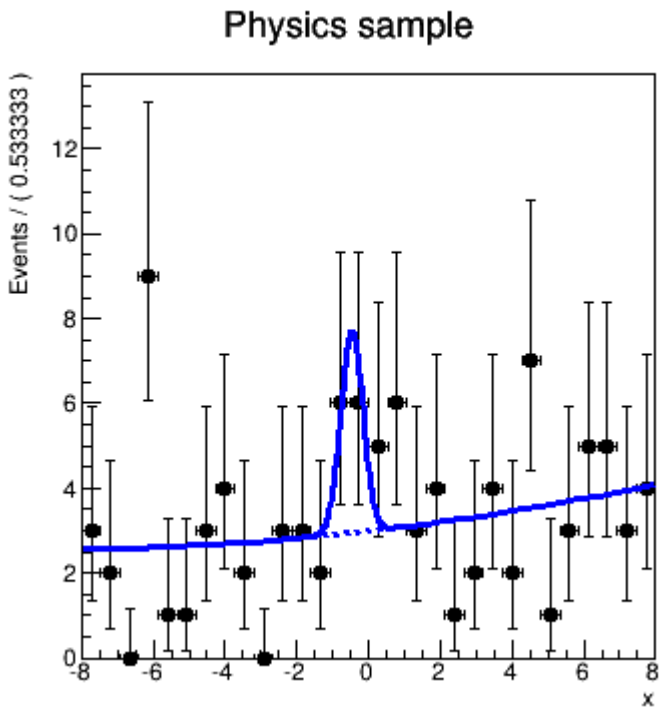
```
printf("Resistant Least trimmed squares fit:\n");
//Now let's try the resistant regression
//The option "rob=0.75" means that we want to use robust fitting and
//we know that at least 75% of data is good points (at least 50% of points
//should be good to use this algorithm). If you don't specify any number
//and just use "rob" for the option, default value of (npoints+nparameters+1)/2
//will be taken
grr->Fit(ffit2, "+rob=0.75");
//
TLegend *leg = new TLegend(0.6, 0.8, 0.89, 0.89);
leg->AddEntry(ffit1, "Ordinary least squares", "l");
leg->AddEntry(ffit2, "LTS regression", "l");
leg->SetFillColor(42);
leg->Draw();

delete [] x;
delete [] y;
delete [] e;
```

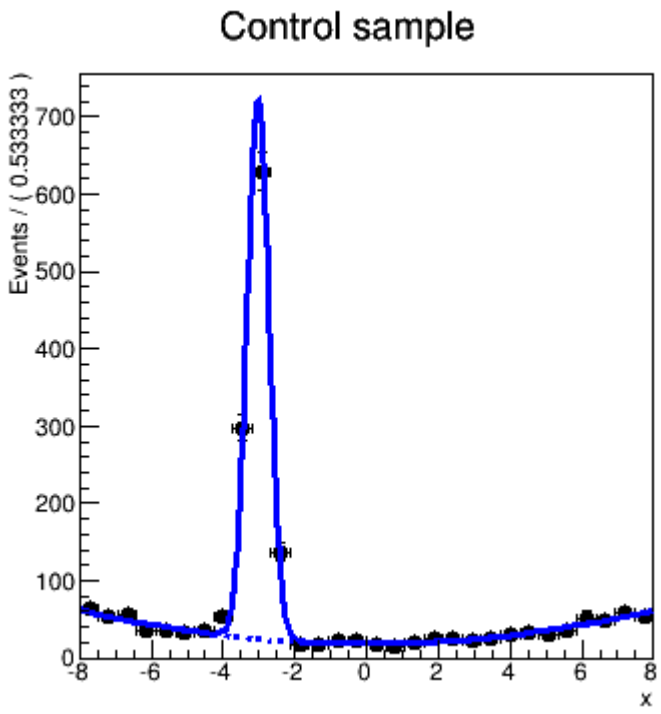
$ROOTSYS/tutorials/fit/fitMultiGraph.C



Siguang@pku.edu.cn

$ROOTSYS/tutorials/roofit/rf501_simultaneouspdf.C





Siguang@pku.edu.cn

$ROOTSYS/tutorials/roofit/rf501_simultaneouspdf.C

```
void rf501_simultaneouspdf()
{
  // C r e a t e   m o d e l   f o r   p h y s i c s   s a m p l e
  // ---------------------------------------------------------

  // Create observables
  RooRealVar x("x","x",-8,8) ;

  // Construct signal pdf
  RooRealVar mean("mean","mean",0,-8,8) ;
  RooRealVar sigma("sigma","sigma",0.3,0.1,10) ;
  RooGaussian gx("gx","gx",x,mean,sigma) ;

  // Construct background pdf
  RooRealVar a0("a0","a0",-0.1,-1,1) ;
  RooRealVar a1("a1","a1",0.004,-1,1) ;
  RooChebychev px("px","px",x,RooArgSet(a0,a1)) ;

  // Construct composite pdf
  RooRealVar f("f","f",0.2,0.,1.) ;
  RooAddPdf model("model","model",RooArgList(gx,px),f) ;
```

$ROOTSYS/tutorials/roofit/rf501_simultaneouspdf.C

```
// C r e a t e   m o d e l   f o r   c o n t r o l   s a m p l e
// ---------------------------------------------------------

// Construct signal pdf.
// NOTE that sigma is shared with the signal sample model
RooRealVar mean_ctl("mean_ctl","mean_ctl",-3,-8,8) ;
RooGaussian gx_ctl("gx_ctl","gx_ctl",x,mean_ctl,sigma) ;

// Construct the background pdf
RooRealVar a0_ctl("a0_ctl","a0_ctl",-0.1,-1,1) ;
RooRealVar a1_ctl("a1_ctl","a1_ctl",0.5,-0.1,1) ;
RooChebychev px_ctl("px_ctl","px_ctl",x,RooArgSet(a0_ctl,a1_ctl)) ;

// Construct the composite model
RooRealVar f_ctl("f_ctl","f_ctl",0.5,0.,1.) ;
RooAddPdf model_ctl("model_ctl","model_ctl",RooArgList(gx_ctl,px_ctl),f_ctl) ;

// C o n s t r u c t   a   s i m u l t a n e o u s   p d f   i n   ( x , s a m p l e )
// -----------------------------------------------------------------------------

// Construct a simultaneous pdf using category sample as index
RooSimultaneous simPdf("simPdf","simultaneous pdf",sample) ;

// Associate model with the physics state and model_ctl with the control state
simPdf.addPdf(model,"physics") ;
simPdf.addPdf(model_ctl,"control") ;
```

共用sigma

$ROOTSYS/tutorials/roofit/rf501_simultaneouspdf.C

```
// G e n e r a t e   e v e n t s   f o r   b o t h   s a m p l e s
// ---------------------------------------------------------------

// Generate 1000 events in x and y from model
RooDataSet *data = model.generate(RooArgSet(x),100) ;
RooDataSet *data_ctl = model_ctl.generate(RooArgSet(x),2000) ;



// C r e a t e   i n d e x   c a t e g o r y   a n d   j o i n   s a m p l e s
// ------------------------------------------------------------------------

// Define category to distinguish physics and control samples events
RooCategory sample("sample","sample") ;
sample.defineType("physics") ;
sample.defineType("control") ;

// Construct combined dataset in (x,sample)
RooDataSet combData("combData","combined data",x,Index(sample),Import("physics",*data),Import("control",*data_ctl)) ;
```

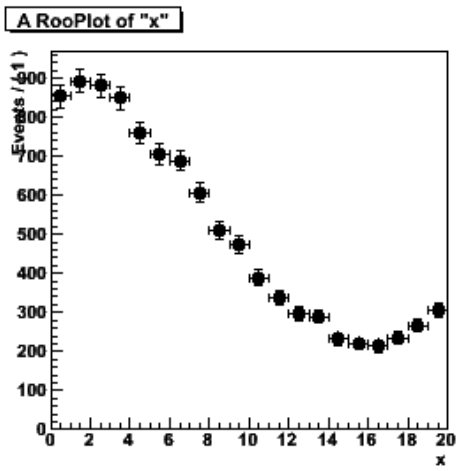
$ROOTSYS/tutorials/roofit/rf501_simultaneouspdf.C

```
// P e r f o r m   a   s i m u l t a n e o u s   f i t
// ---------------------------------------------------

// Perform simultaneous fit of model to data and model_ctl to data_ctl
simPdf.fitTo(combData) ;



// P l o t   m o d e l   s l i c e s   o n   d a t a   s l i c e s
// ------------------------------------------------------------

// Make a frame for the physics sample
RooPlot* frame1 = x.frame(Bins(30),Title("Physics sample")) ;


// Plot all data tagged as physics sample
combData.plotOn(frame1,Cut("sample==sample::physics")) ;

// Plot "physics" slice of simultaneous pdf.
// NBL You _must_ project the sample index category with data using ProjWData
// as a RooSimultaneous makes no prediction on the shape in the index category
// and can thus not be integrated
simPdf.plotOn(frame1,Slice(sample,"physics"),ProjWData(sample,combData)) ;
simPdf.plotOn(frame1,Slice(sample,"physics"),Components("px"),ProjWData(sample,combData),LineStyle(kDashed)) ;

// The same plot for the control sample slice
RooPlot* frame2 = x.frame(Bins(30),Title("Control sample")) ;
combData.plotOn(frame2,Cut("sample==sample::control")) ;
simPdf.plotOn(frame2,Slice(sample,"control"),ProjWData(sample,combData)) ;
simPdf.plotOn(frame2,Slice(sample,"control"),Components("px_ctl"),ProjWData(sample,combData),LineStyle(kDashed)) ;


 TCanvas* c = new TCanvas("rf501_simultaneouspdf","rf403_simultaneouspdf",800,400) ;
 c->Divide(2) ;
 c->cd(1) ; gPad->SetLeftMargin(0.15) ; frame1->GetYaxis()->SetTitleOffset(1.4) ; frame1->Draw() ;
 c->cd(2) ; gPad->SetLeftMargin(0.15) ; frame2->GetYaxis()->SetTitleOffset(1.4) ; frame2->Draw() ;
```

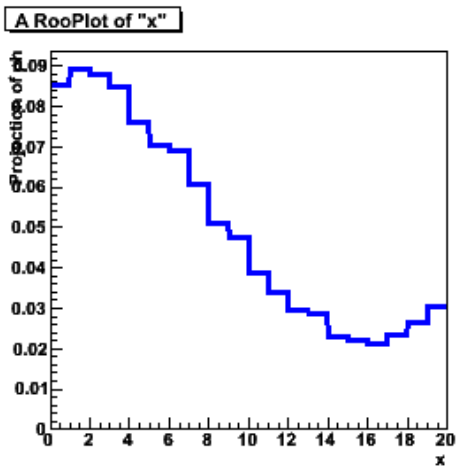
Highlight of non-parametric shapes - histograms

- Class **RooHistPdf** – a p.d.f. described by a histogram

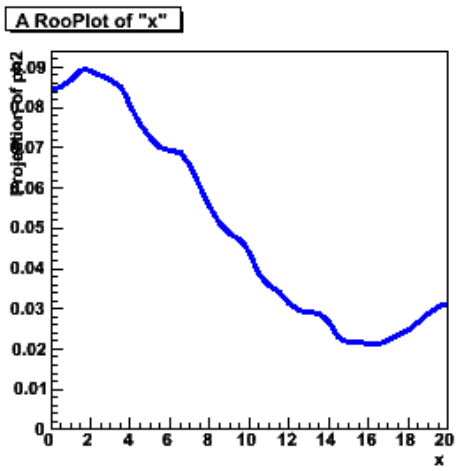
dataHist



RooHistPdf(N=0)



RooHistPdf(N=4)

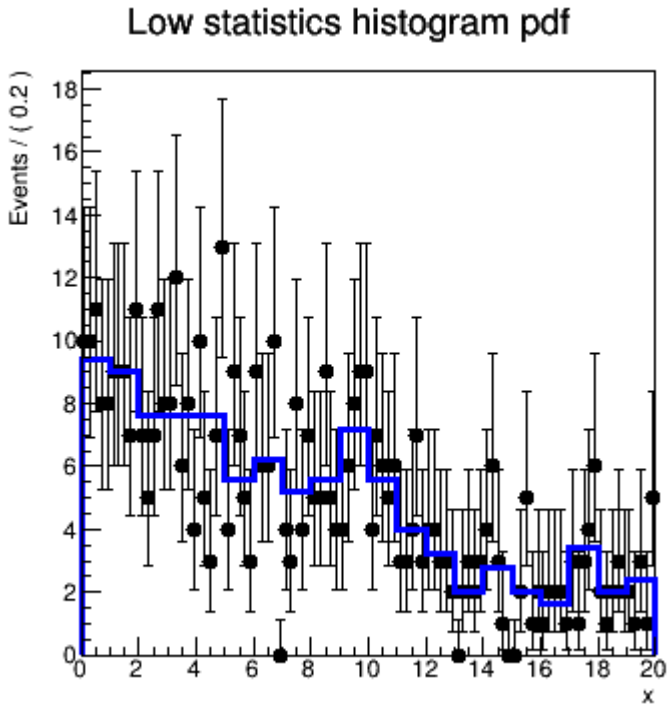


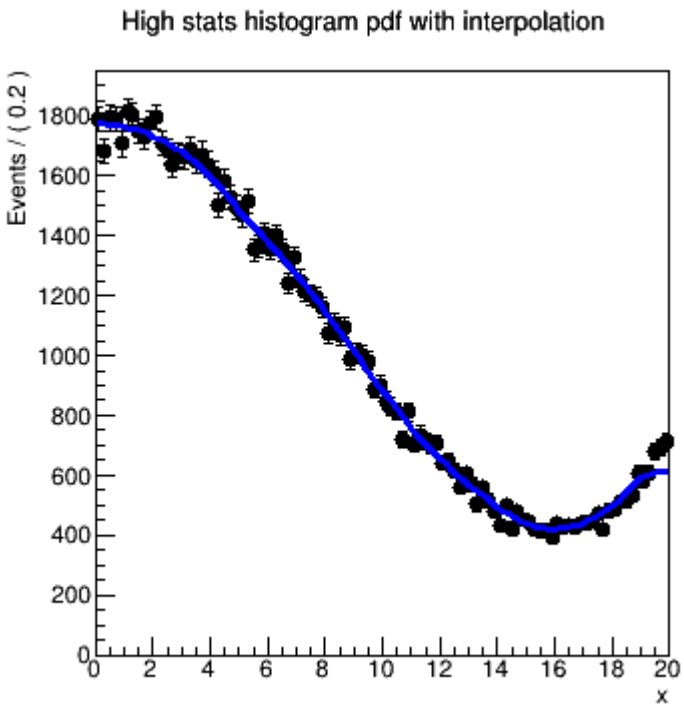

```
// Histogram based p.d.f with N-th order interpolation
RooHistPdf ph("ph","ph",x,*dataHist,N) ;
```

- Not so great at low statistics (especially problematic in >1 dim)

Siguang@pku.edu.cn

RooHistPdf





$ROOTSYS/tutorials/roofit/rf706_histpdf.C

$ROOTSYS/tutorials/roofit/rf706_histpdf.C

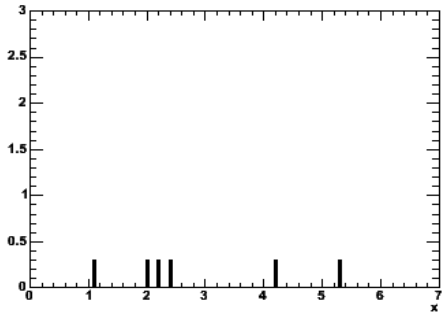
```
void rf706_histpdf()
{
  // C r e a t e   p d f   f o r   s a m p l i n g
  // ---------------------------------------------

  RooRealVar x("x","x",0,20) ;
  RooPolynomial p("p","p",x,RooArgList(RooConst(0.01),RooConst(-0.01),RooConst(0.0004))) ;



  // C r e a t e   l o w   s t a t s   h i s t o g r a m
  // ---------------------------------------------------

  // Sample 500 events from p
  x.setBins(20) ;
  RooDataSet* data1 = p.generate(x,500) ;

  // Create a binned dataset with 20 bins and 500 events
  RooDataHist* hist1 = data1->binnedClone() ;

  // Represent data in dh as pdf in x
  RooHistPdf histpdf1("histpdf1","histpdf1",x,*hist1,0) ;

  // Plot unbinned data and histogram pdf overlaid
  RooPlot* frame1 = x.frame(Title("Low statistics histogram pdf"),Bins(100)) ;
  data1->plotOn(frame1) ;
  histpdf1.plotOn(frame1) ;
```

$ROOTSYS/tutorials/roofit/rf706_histpdf.C

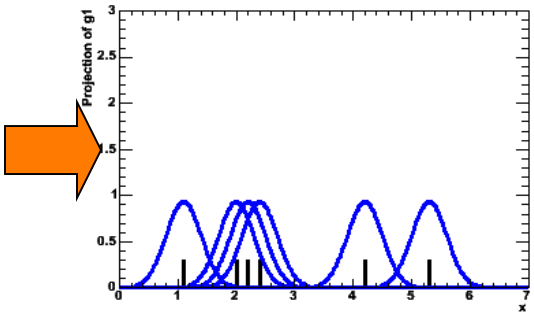
```
  // C r e a t e   h i g h   s t a t s   h i s t o g r a m
  // ---------------------------------------------------

  // Sample 100000 events from p
  x.setBins(10) ;
  RooDataSet* data2 = p.generate(x,100000) ;

  // Create a binned dataset with 10 bins and 100K events
  RooDataHist* hist2 = data2->binnedClone() ;

  // Represent data in dh as pdf in x, apply 2nd order interpolation
  RooHistPdf histpdf2("histpdf2","histpdf2",x,*hist2,2) ;

  // Plot unbinned data and histogram pdf overlaid
  RooPlot* frame2 = x.frame(Title("High stats histogram pdf with interpolation"),Bins(100)) ;
  data2->plotOn(frame2) ;
  histpdf2.plotOn(frame2) ;


  TCanvas* c = new TCanvas("rf706_histpdf","rf706_histpdf",800,400) ;
  c->Divide(2) ;
  c->cd(1) ; gPad->SetLeftMargin(0.15) ; frame1->GetYaxis()->SetTitleOffset(1.4) ; frame1->Draw() ;
  c->cd(2) ; gPad->SetLeftMargin(0.15) ; frame2->GetYaxis()->SetTitleOffset(1.8) ; frame2->Draw() ;


}
```

- Class `RooKeysPdf` – A kernel estimation p.d.f.
  - Uses *unbinned* data
  - Idea represent each event of your MC sample as a Gaussian probability distribution
  - Add probability distributions from all events in sample

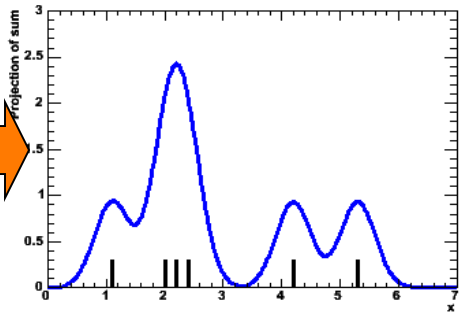
Sample of events



Gaussian probability distributions for each event



Summed probability distribution for all events in sample

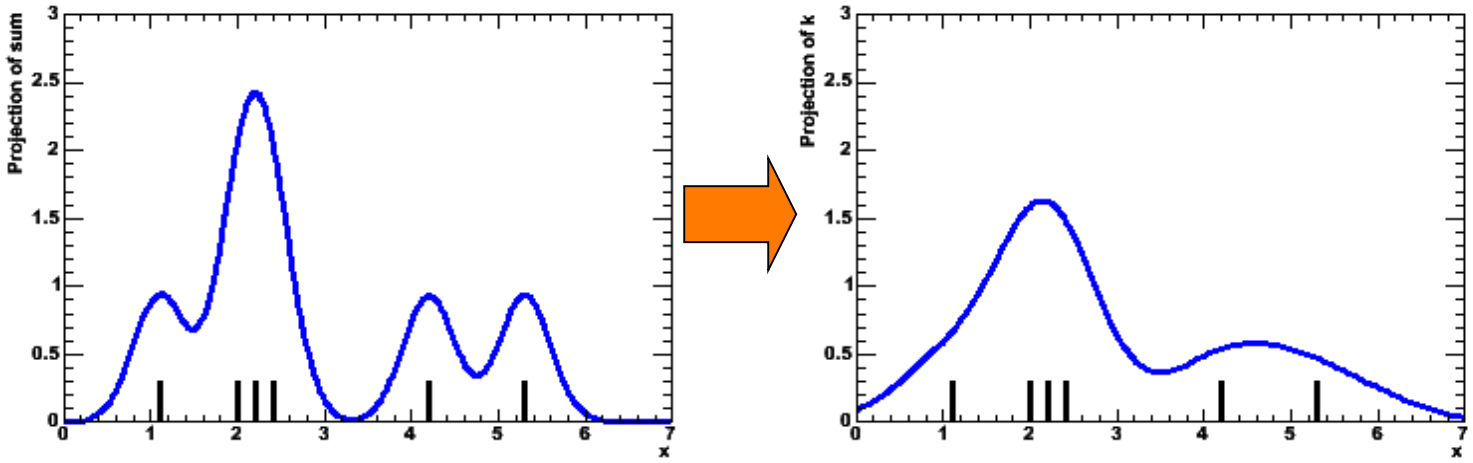


Highlight of non-parametric shapes – kernel estimation

- Width of Gaussian kernels need not be the same for all events
  - As long as each event contributes 1/N to the integral
- Idea: 'Adaptive kernel' technique
  - Choose wide Gaussian if local density of events is low
  - Choose narrow Gaussian if local density of events is high
  - Preserves small features in high statistics areas, minimize jitter in low statistics areas
  - Automatically calculated

Static Kernel
(with of all Gaussian identical)

Adaptive Kernel
(width of all Gaussian depends on local density of events)



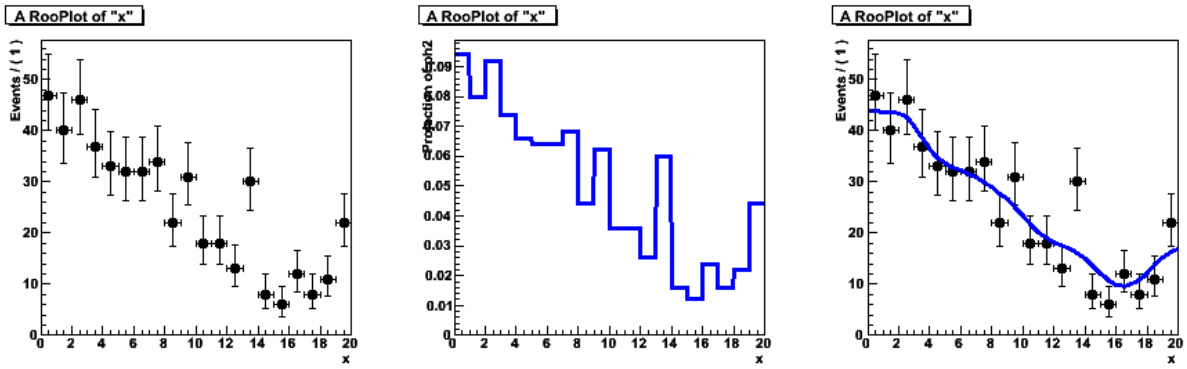
Siguang@pku.edu.cn

Highlight of non-parametric shapes – kernel estimation

```
// Adaptive kernel estimation p.d.f
 RooKeysPdf k("k","k",x,*d,RooKeysPdf::MirrorBoth) ;
```

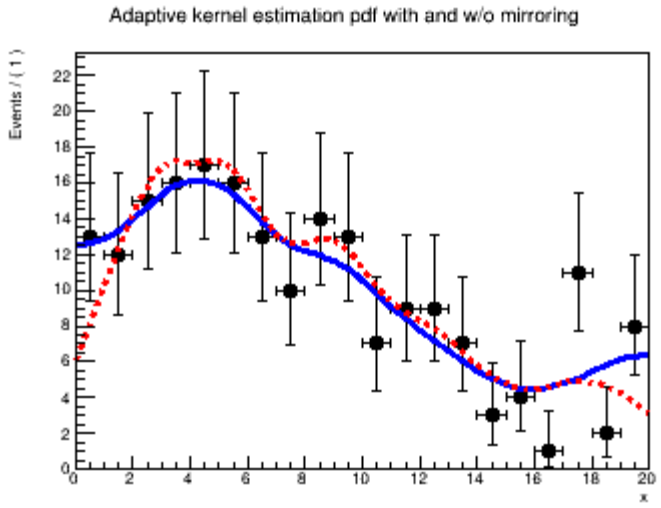
- Example with comparison to histogram based p.d.f
  - Superior performance at low statistics
  - Can mirror input data over boundaries to reduce 'edge leakage'
  - Works also in >1 dimensions (class **RooNDKeysPdf**)

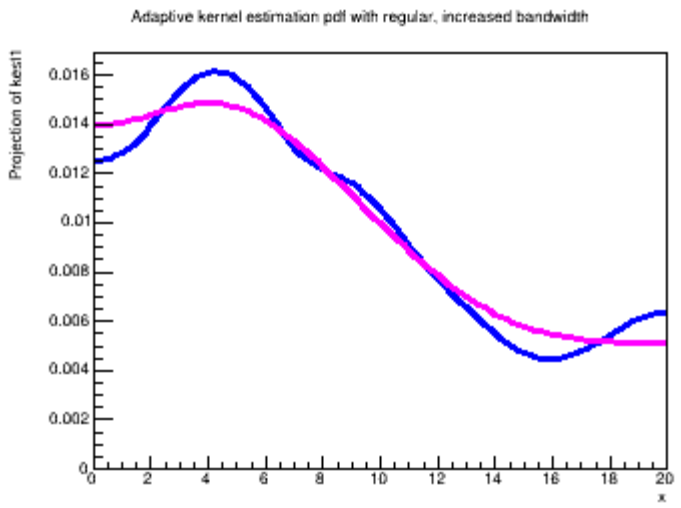
Data (N=500) RooHistPdf(data) RooKeysPdf(data)

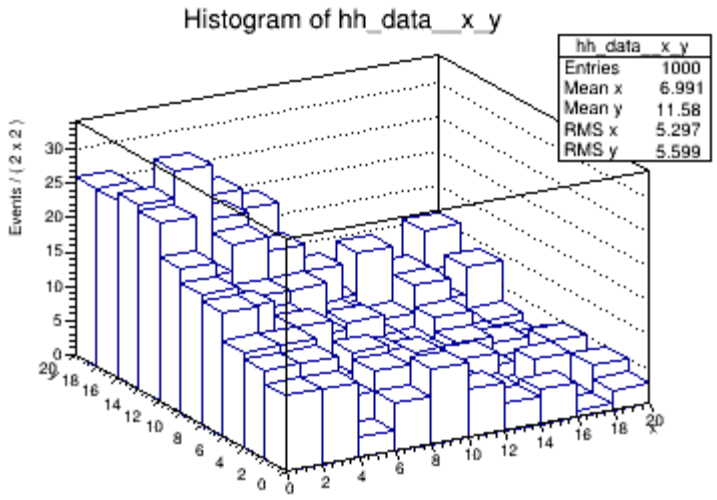


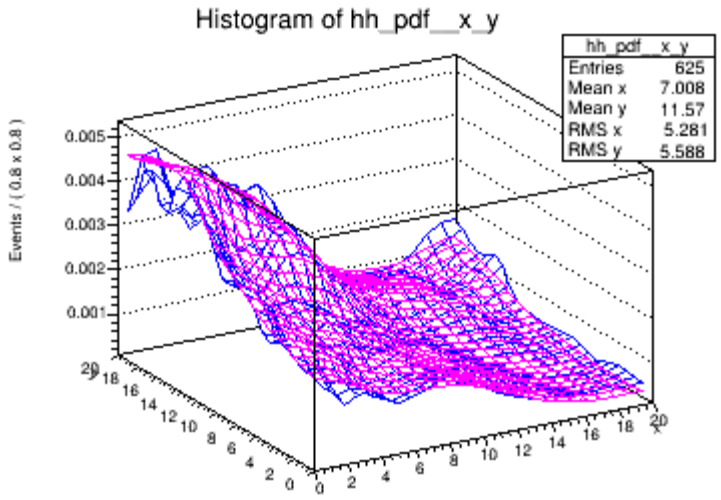
Siguang@pku.edu.cn

$ROOTSYS/tutorials/roofit/rf707_kernelestimation.C









$ROOTSYS/tutorials/roofit/rf707_kernelestimation.C

```
void rf707_kernelestimation()
{
  // C r e a t e   l o w   s t a t s   1 - D   d a t a s e t
  // ---------------------------------------------------

  // Create a toy pdf for sampling
  RooRealVar x("x","x",0,20) ;
  RooPolynomial p("p","p",x,RooArgList(RooConst(0.01),RooConst(-0.01),RooConst(0.0004))) ;

  // Sample 500 events from p
  RooDataSet* data1 = p.generate(x,200) ;



  // C r e a t e   1 - D   k e r n e l   e s t i m a t i o n   p d f
  // -------------------------------------------------------------

  // Create adaptive kernel estimation pdf. In this configuration the input data
  // is mirrored over the boundaries to minimize edge effects in distribution
  // that do not fall to zero towards the edges
  RooKeysPdf kest1("kest1","kest1",x,*data1,RooKeysPdf::MirrorBoth) ;

  // An adaptive kernel estimation pdf on the same data without mirroring option
  // for comparison
  RooKeysPdf kest2("kest2","kest2",x,*data1,RooKeysPdf::NoMirror) ;
```

$ROOTSYS/tutorials/roofit/rf707_kernelestimation.C

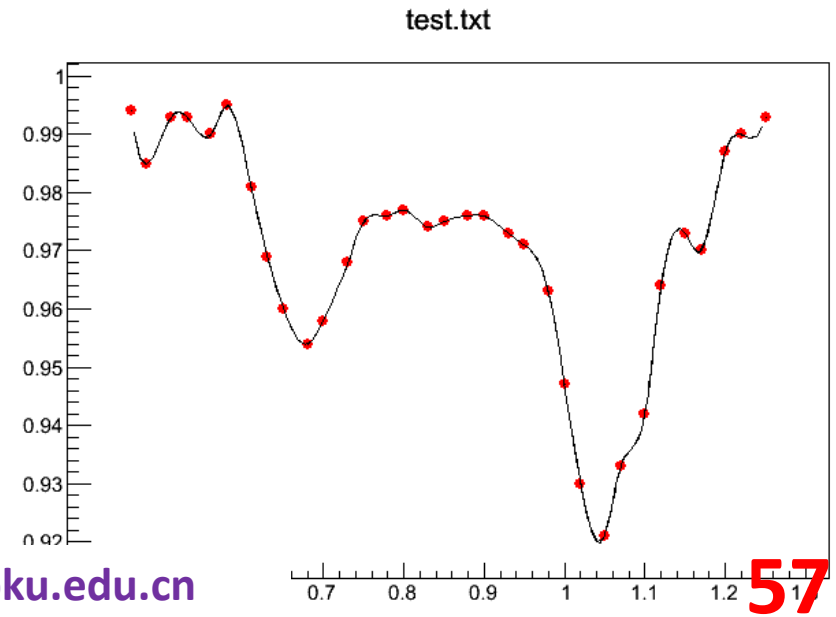
```
// Adaptive kernel estimation pdf with increased bandwidth scale factor
// (promotes smoothness over detail preservation)
RooKeysPdf kest3("kest1","kest1",x,*data1,RooKeysPdf::MirrorBoth,2) ;


// Plot kernel estimation pdfs with and without mirroring over data
RooPlot* frame = x.frame(Title("Adaptive kernel estimation pdf with and w/o mirroring"),Bins(20)) ;
data1->plotOn(frame) ;
kest1.plotOn(frame) ;
kest2.plotOn(frame,LineStyle(kDashed),LineColor(kRed)) ;


// Plot kernel estimation pdfs with regular and increased bandwidth
RooPlot* frame2 = x.frame(Title("Adaptive kernel estimation pdf with regular, increased bandwidth")) ;
kest1.plotOn(frame2) ;
kest3.plotOn(frame2,LineColor(kMagenta)) ;
```

.....

TSpline

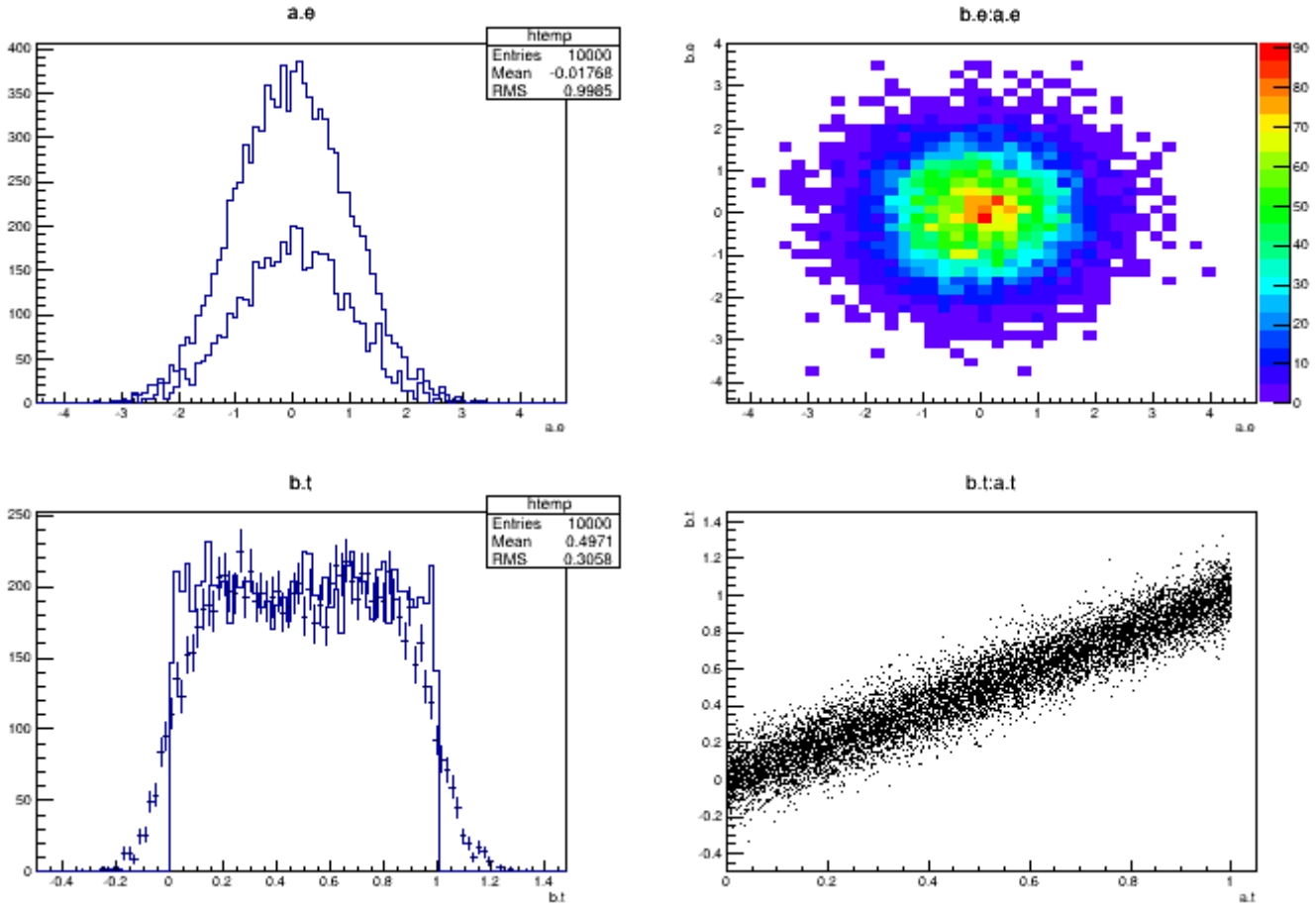
```
void test(){
  TGraph *gr = new TGraph("test.txt");
  gr->SetMarkerStyle(20);
  gr->SetMarkerColor(kRed);
  gr->Draw("AP");
  TSpline3 *sp = new TSpline3("sp",gr);
  sp->Draw("lcpsame");
  for(Int_t i=0; i<10; i++){
    Double_t x = 0.4+i*(1.3-0.4)/10.;
    printf("x=%f Eff = %f\n",x,sp->Eval(x));
  }
}
```

0.46	0.994
0.48	0.985
0.51	0.993
0.53	0.993
0.56	0.99
0.58	0.995
0.61	0.981
0.63	0.969
0.65	0.96
0.68	0.954



Siguang@pku.edu.cn

$ROOTSYS/tutorials/tree/tree0.C



Siguang@pku.edu.cn

$ROOTSYS/tutorials/tree/tree0.C

```
#include <TRandom.h>
#include <TTree.h>
#include <TCanvas.h>
#include <TStyle.h>

#include <Riostream.h>

//class Det   : public TObject  {
class Det {  // each detector gives an energy and time signal
public:
  Double_t e; //energy
  Double_t t; //time

//  ClassDef(Det,1)
};

//ClassImp(Det)

//class Event { //TObject is not required by this example
class Event : public TObject {
public:

  Det a; // say there are two detectors (a and b) in the experiment
  Det b;
  ClassDef(Event,1)
};

ClassImp(Event)
```

$ROOTSYS/tutorials/tree/tree0.C

```
void tree0() {
  // create a TTree
  TTree *tree = new TTree("tree","treelibrated tree");
  Event *e = new Event;

  // create a branch with energy
  tree->Branch("event",&e);

  // fill some events with random numbers
  Int_t nevent=10000;
  for (Int_t iev=0;iev<nevent;iev++) {
    if (iev%1000==0) cout<<"Processing event "<<iev<<"..."<<endl;

    Float_t ea,eb;
    gRandom->Rannor(ea,eb); // the two energies follow a gaus distribution
    e->a.e=ea;
    e->b.e=eb;
    e->a.t=gRandom->Rndm();  // random
    e->b.t=e->a.t + gRandom->Gaus(0.,.1);  // identical to a.t but a gaussian
                                           // 'resolution' was added with sigma .1

    tree->Fill();  // fill the tree with the current event
  }

  // start the viewer
  // here you can investigate the structure of your Event class
  tree->StartViewer();

  //gROOT->SetStyle("Plain");   // uncomment to set a different style
  gStyle->SetPalette(1);        // use precomputed color palette 1
```

$ROOTSYS/tutorials/tree/tree0.C

```
  // now draw some tree variables
  TCanvas *c1 = new TCanvas();
  c1->Divide(2,2);
  c1->cd(1);
  tree->Draw("a.e");  //energy of det a
  tree->Draw("a.e","3*(-.2<b.e && b.e<.2)","same");  // same but with condition on energy b; scaled by 3
  c1->cd(2);
  tree->Draw("b.e:a.e","","colz");        // one energy against the other
  c1->cd(3);
  tree->Draw("b.t","","e");    // time of b with errorbars
  tree->Draw("a.t","","same"); // overlay time of detector a
  c1->cd(4);
  tree->Draw("b.t:a.t");       // plot time b again time a

  cout<<endl;
  cout<<"You can now examine the structure of your tree in the TreeViewer"<<endl;
  cout<<endl;
}
```

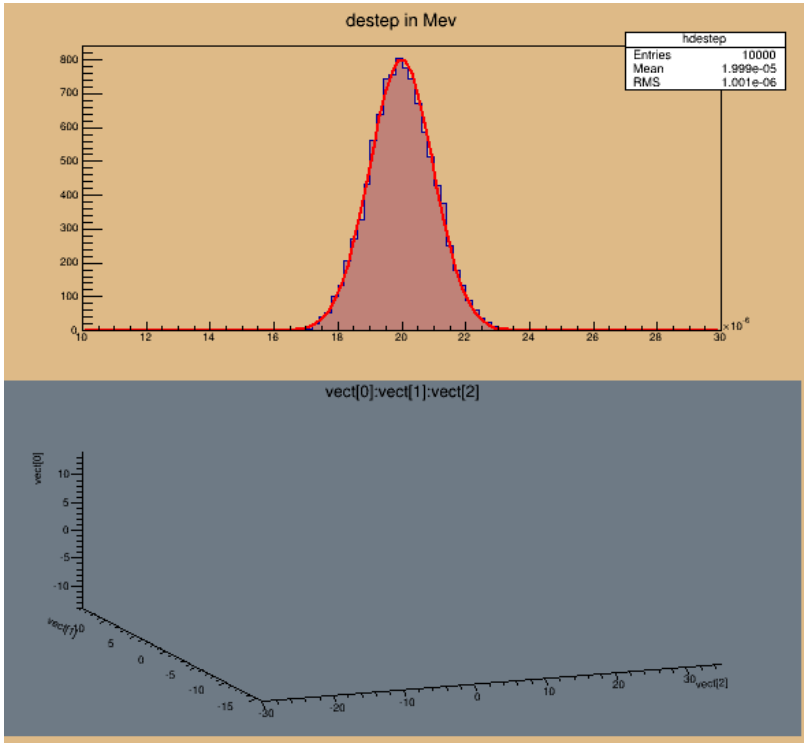
$ROOTSYS/tutorials/tree/tree1.C

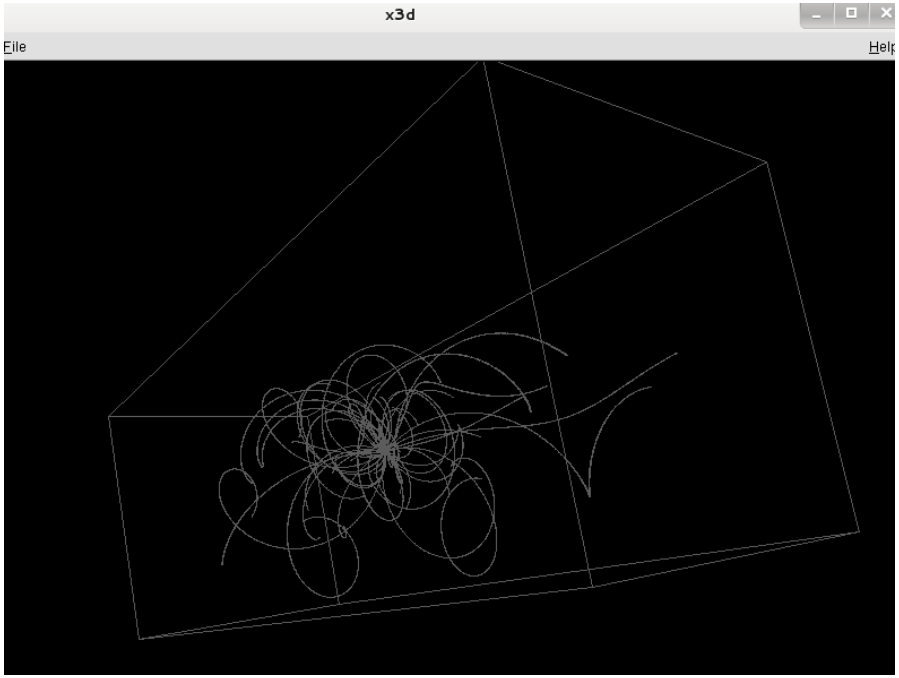
```
void tree1w()
{
   //create a Tree file tree1.root

   //create the file, the Tree and a few branches
   TFile f("tree1.root","recreate");
   TTree t1("t1","a simple Tree with simple variables");
   Float_t px, py, pz;
   Double_t random;
   Int_t ev;
   t1.Branch("px",&px,"px/F");
   t1.Branch("py",&py,"py/F");
   t1.Branch("pz",&pz,"pz/F");
   t1.Branch("random",&random,"random/D");
   t1.Branch("ev",&ev,"ev/I");

   //fill the tree
   for (Int_t i=0;i<10000;i++) {
     gRandom->Rannor(px,py);
     pz = px*px + py*py;
     random = gRandom->Rndm();
     ev = i;
     t1.Fill();
  }

  //save the Tree header. The file will be automatically closed
  //when going out of the function scope
  t1.Write();
}
```

$ROOTSYS/tutorials/tree/tree1.C

```
void tree1r()
{
   //read the Tree generated by tree1w and fill two histograms

   //note that we use "new" to create the TFile and TTree objects !
   //because we want to keep these objects alive when we leave this function.
   TFile *f = new TFile("tree1.root");
   TTree *t1 = (TTree*)f->Get("t1");
   Float_t px, py, pz;
   Double_t random;
   Int_t ev;
   t1->SetBranchAddress("px",&px);
   t1->SetBranchAddress("py",&py);
   t1->SetBranchAddress("pz",&pz);
   t1->SetBranchAddress("random",&random);
   t1->SetBranchAddress("ev",&ev);

   //create two histograms
   TH1F *hpx   = new TH1F("hpx","px distribution",100,-3,3);
   TH2F *hpxpy = new TH2F("hpxpy","py vs px",30,-3,3,30,-3,3);

   //read all entries and fill the histograms
   Long64_t nentries = t1->GetEntries();
   for (Long64_t i=0;i<nentries;i++) {
     t1->GetEntry(i);
     hpx->Fill(px);
     hpxpy->Fill(px,py);
  }

  //we do not close the file. We want to keep the generated histograms
  //we open a browser and the TreeViewer
  if (gROOT->IsBatch()) return;
  new TBrowser();
  t1->StartViewer();
  // in the browser, click on "ROOT Files", then on "tree1.root".
  //     you can click on the histogram icons in the right panel to draw them.
  // in the TreeViewer, follow the instructions in the Help button.
}
```

$ROOTSYS/tutorials/tree/tree2.C





$ROOTSYS/tutorials/tree/tree2.C

```
const Int_t MAXMEC = 30;

class Gctrak : public TObject {
public:
  Float_t  vect[7];
  Float_t  getot;
  Float_t  gekin;
  Float_t  vout[7];   //! not persistent
  Int_t    nmec;
  Int_t   *lmec;      //[nmec]
  Int_t   *namec;     //[nmec]
  Int_t    nstep;     //! not persistent
  Int_t    pid;
  Float_t  destep;
  Float_t  destel;    //! not persistent
  Float_t  safety;    //! not persistent
  Float_t  sleng;     //! not persistent
  Float_t  step;      //! not persistent
  Float_t  snext;     //! not persistent
  Float_t  sfield;    //! not persistent
  Float_t  tofg;      //! not persistent
  Float_t  gekrat;    //! not persistent
  Float_t  upwght;    //! not persistent

  Gctrak() {lmec=0; namec=0;}

  ClassDef(Gctrak,1)
};
```

$ROOTSYS/tutorials/tree/tree2.C

```
void helixStep(Float_t step, Float_t *vect, Float_t *vout)
{
  // extrapolate track in constant field
   Float_t field = 20;       //magnetic field in kilogauss
   enum Evect {kX,kY,kZ,kPX,kPY,kPZ,kPP};
   vout[kPP] = vect[kPP];
   Float_t h4    = field*2.99792e-4;
   Float_t rho   = -h4/vect[kPP];
   Float_t tet   = rho*step;
   Float_t tsint = tet*tet/6;
   Float_t sintt = 1 - tsint;
   Float_t sint  = tet*sintt;
   Float_t cos1t = tet/2;
   Float_t f1 = step*sintt;
   Float_t f2 = step*cos1t;
   Float_t f3 = step*tsint*vect[kPZ];
   Float_t f4 = -tet*cos1t;
   Float_t f5 = sint;
   Float_t f6 = tet*cos1t*vect[kPZ];
   vout[kX]   = vect[kX]  + (f1*vect[kPX] - f2*vect[kPY]);
   vout[kY]   = vect[kY]  + (f1*vect[kPY] + f2*vect[kPX]);
   vout[kZ]   = vect[kZ]  + (f1*vect[kPZ] + f3);
   vout[kPX]  = vect[kPX] + (f4*vect[kPX] - f5*vect[kPY]);
   vout[kPY]  = vect[kPY] + (f4*vect[kPY] + f5*vect[kPX]);
   vout[kPZ]  = vect[kPZ] + (f4*vect[kPZ] + f6);
}
```

$ROOTSYS/tutorials/tree/tree2.C

```
void tree2aw()
{
   //create a Tree file tree2.root

   //create the file, the Tree and a few branches with
   //a subset of gctrak
   TFile f("tree2.root","recreate");
   TTree t2("t2","a Tree with data from a fake Geant3");
   Gctrak *gstep = new Gctrak;
   t2.Branch("track",&gstep,8000,1);

   //Initialize particle parameters at first point
   Float_t px,py,pz,p,charge=0;
   Float_t vout[7];
   Float_t mass  = 0.137;
   Bool_t newParticle = kTRUE;
   gstep->lmec    = new Int_t[MAXMEC];
   gstep->namec   = new Int_t[MAXMEC];
   gstep->step    = 0.1;
   gstep->destep  = 0;
   gstep->nmec    = 0;
   gstep->pid     = 0;
```

$ROOTSYS/tutorials/tree/tree2.C

```
//transport particles
for (Int_t i=0;i<10000;i++) {
   //generate a new particle if necessary
   if (newParticle) {
      px = gRandom->Gaus(0,.02);
      py = gRandom->Gaus(0,.02);
      pz = gRandom->Gaus(0,.02);
      p  = TMath::Sqrt(px*px+py*py+pz*pz);
      charge = 1; if (gRandom->Rndm() < 0.5) charge = -1;
      gstep->pid    += 1;
      gstep->vect[0] = 0;
      gstep->vect[1] = 0;
      gstep->vect[2] = 0;
      gstep->vect[3] = px/p;
      gstep->vect[4] = py/p;
      gstep->vect[5] = pz/p;
      gstep->vect[6] = p*charge;
      gstep->getot   = TMath::Sqrt(p*p + mass*mass);
      gstep->gekin   = gstep->getot - mass;
      newParticle = kFALSE;
   }

   // fill the Tree with current step parameters
   t2.Fill();

   //transport particle in magnetic field
   helixStep(gstep->step, gstep->vect, vout); //make one step
```

$ROOTSYS/tutorials/tree/tree2.C

```
      //apply energy loss
      gstep->destep = gstep->step*gRandom->Gaus(0.0002,0.00001);
      gstep->gekin -= gstep->destep;
      gstep->getot  = gstep->gekin + mass;
      gstep->vect[6] = charge*TMath::Sqrt(gstep->getot*gstep->getot - mass*mass
;
      gstep->vect[0] = vout[0];
      gstep->vect[1] = vout[1];
      gstep->vect[2] = vout[2];
      gstep->vect[3] = vout[3];
      gstep->vect[4] = vout[4];
      gstep->vect[5] = vout[5];
      gstep->nmec    = (Int_t)(5*gRandom->Rndm());
      for (Int_t l=0;l<gstep->nmec;l++) {
         gstep->lmec[l] = l;
         gstep->namec[l] = l+100;
      }
      if (gstep->gekin < 0.001)            newParticle = kTRUE;
      if (TMath::Abs(gstep->vect[2]) > 30) newParticle = kTRUE;
   }

   //save the Tree header. The file will be automatically closed
   //when going out of the function scope
   t2.Write();
}
```

$ROOTSYS/tutorials/tree/tree2.C

```
void tree2ar()
{
   //read the Tree generated by tree2w and fill one histogram
   //we are only interested by the destep branch.

   //note that we use "new" to create the TFile and TTree objects !
   //because we want to keep these objects alive when we leave
   //this function.
   TFile *f = new TFile("tree2.root");
   TTree *t2 = (TTree*)f->Get("t2");
   Gctrak *gstep = 0;
   t2->SetBranchAddress("track",&gstep);
   TBranch *b_destep = t2->GetBranch("destep");

   //create one histogram
   TH1F *hdestep   = new TH1F("hdestep","destep in Mev",100,1e-5,3e-5);

   //read only the destep branch for all entries
   Long64_t nentries = t2->GetEntries();
   for (Long64_t i=0;i<nentries;i++) {
      b_destep->GetEntry(i);
      hdestep->Fill(gstep->destep);
   }
```

$ROOTSYS/tutorials/tree/tree2.C

```
   //we do not close the file.
   //We want to keep the generated histograms
   //We fill a 3-d scatter plot with the particle step coordinates
   TCanvas *c1 = new TCanvas("c1","c1",600,800);
   c1->SetFillColor(42);
   c1->Divide(1,2);
   c1->cd(1);
   hdestep->SetFillColor(45);
   hdestep->Fit("gaus");
   c1->cd(2);
   gPad->SetFillColor(37);
   t2->SetMarkerColor(kRed);
   t2->Draw("vect[0]:vect[1]:vect[2]");
   if (gROOT->IsBatch()) return;

   // invoke the x3d viewer
   gPad->GetViewer3D("x3d");
}

void tree2a() {
   tree2aw();
   tree2ar();
}
```

ROOT的安装

ROOT的安装

快速安装：在联网的状态下，在Ubuntu、Debian操作系统下执行

apt-get install root-system

即可

编译源代码安装

从root.cern.ch获取源代码后,执行如下代码解压缩

tar –zxvf root root_v5.34.18.source.tar.gz

执行结果是在当前目录下产生一root子目录，然后顺序执行如下：

```
#install root under debian----
export ROOTSYS=/home/wsg/work/root/534/
mkdir -p $ROOTSYS
cd root
#fftw
apt-get install fftw3

#For Debian to compile
apt-get install build-essential
apt-get instal ldpkg-dev
#OpenGL
apt-get install libxft-dev
apt-get install libgl1-mesa-dev
apt-get install libxt-dev
apt-get install libxmu-dev
apt-get install libxmu-headers
apt-get install libxmu6
apt-get install libxmu6-dbg
apt-get install libxmuu-dev
apt-get install libxmuu1
apt-get install  libxmuu1-dbg &

#
apt-get intall libghc-sdl-ttf-dev &

cmake ../root -DCMAKE_INSTALL_PREFIX=~/work/root/534 -Droottest=ON -Dfftw3=ON -Droofit=ON -Dqt=ON
make -j2
make install
```

注：不同环境已经安装的包不同，在运行cmake时，如果缺XXX库什么系统会提醒，用apt-cache search XXX查找，用apt-get install XXX 安装

编译源代码安装(更省事的安装)

```
export FFTW_DIR=/scratch/other/wangsg/root64/fftw
mkdir tmpRootCompile
cd tmpRootCompile
cmake ../root -DCMAKE_INSTALL_PREFIX =/home/wsg/work/root/534 -Dall=on  -Dfail-on-missing=OFF

make -j40
make install
```

其中:
../root 指向root解压缩后的代码目录
-DCMAKE_INSTALL_PREFIX =/home/wsg/work/root/534指向安装目录
-Dall=on 打开所有选项
-Dfail-on-missing=OFF 如果没有找到需要的外挂库,继续执行其余安装
make –j40 用40个核同时编译
make install 进行安装

设置环境

检查运行环境： echo $0

如果返回bash

```
wsg@debian:~$ cd
wsg@debian:~$ emacs .bashrc &
在文件中加入：
export ROOTSYS=/home/wsg/work/root/534
export PATH=$ROOTSYS/bin:$PATH
export LD_LIBRARY_PATH=$ROOTSYS/lib:$LD_LIBRARY_PATH
```

如果返回-tcsh

```
wsg@debian:~$ cd
wsg@debian:~$ emacs .tcshrc &
在文件中加入：
setenv ROOTSYS   /home/wsg/work/root/534
setenv PATH   $ROOTSYS/bin:$PATH
setenv LD_LIBRARY_PATH   $ROOTSYS/lib:$LD_LIBRARY_PATH
```

重新开窗口即可输入root

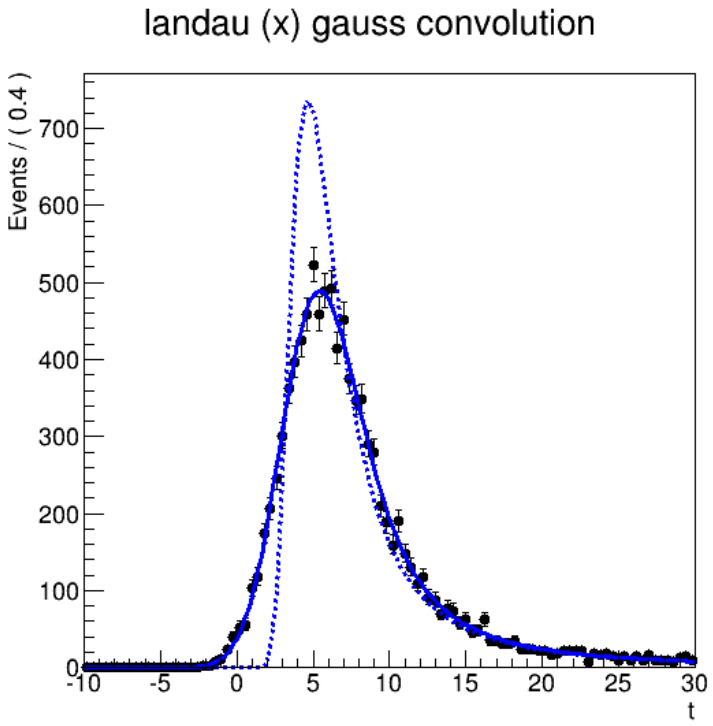
安装后的运行

$ROOTSYS/tutorials/

在$ROOTSYS/tutorials/下有很多例子程序，运行方法为：

$cd $ROOTSYS/tutorials/roofit

$root rf208_convolution.C



看能否出现左图。
如果能，说明你的root、roofit软件包、fftw软件安装成功。

如果不能。。。。。或许系统差异大。。。。直接安装虚拟机好了。

Siguang@pku.edu.cn

编译运行

红色为您输入的文字

```
wsg@debian:~/work/root/534/tutorials/roofit$ root
root [0] .L rf208_convolution.C++
Info in <TUnixSystem::ACLiC>: creating shared library
/home/wsg/work/root/534/tutorials/roofit/./rf208_convolution_C.so

RooFit v3.60 -- Developed by Wouter Verkerke and David Kirkby
                Copyright (C) 2000-2013 NIKHEF, University of California & Stanford University
                All rights reserved, please read http://roofit.sourceforge.net/license.txt

root [1] rf208_convolution()
```

rf208_convolution.C

编译运行头文件要包含

```
///////////////////////////////////////////////////////////////////////
//
// 'ADDITION AND CONVOLUTION' RooFit tutorial macro #208
//
// One-dimensional numeric convolution
// (require ROOT to be compiled with --enable-fftw3)
//
// pdf = landau(t) (x) gauss(t)
//
//
// 07/2008 - Wouter Verkerke
//
///////////////////////////////////////////////////////////////////////

#ifndef __CINT__
#include "RooGlobalFunc.h"
#endif
#include "RooRealVar.h"
#include "RooDataSet.h"
#include "RooGaussian.h"
#include "RooLandau.h"
#include "RooFFTConvPdf.h"
#include "RooPlot.h"
#include "TCanvas.h"
#include "TAxis.h"
#include "TH1.h"
using namespace RooFit ;
```

rf208_convolution.C

```
void rf208_convolution()
{
  // S e t u p   c o m p o n e n t   p d f s
  // ---------------------------------------

  // Construct observable
  RooRealVar t("t","t",-10,30) ;

  // Construct landau(t,ml,sl) ;
  RooRealVar ml("ml","mean landau",5.,-20,20) ;
  RooRealVar sl("sl","sigma landau",1,0.1,10) ;
  RooLandau landau("lx","lx",t,ml,sl) ;

  // Construct gauss(t,mg,sg)
  RooRealVar mg("mg","mg",0) ;
  RooRealVar sg("sg","sg",2,0.1,10) ;
  RooGaussian gauss("gauss","gauss",t,mg,sg) ;


  // C o n s t r u c t   c o n v o l u t i o n   p d f
  // -------------------------------------------------

  // Set #bins to be used for FFT sampling to 10000
  t.setBins(10000,"cache") ;

  // Construct landau (x) gauss
  RooFFTConvPdf lxg("lxg","landau (X) gauss",t,landau,gauss) ;
```

rf208_convolution.C

```
  // S a m p l e ,   f i t   a n d   p l o t   c o n v o l u t e d   p d f
  // ---------------------------------------------------------------------

  // Sample 1000 events in x from gxlx
  RooDataSet* data = lxg.generate(t,10000) ;

  // Fit gxlx to data
  lxg.fitTo(*data) ;

  // Plot data, landau pdf, landau (X) gauss pdf
  RooPlot* frame = t.frame(Title("landau (x) gauss convolution")) ;
  data->plotOn(frame) ;
  lxg.plotOn(frame) ;
  landau.plotOn(frame,LineStyle(kDashed)) ;


  // Draw frame on canvas
  new TCanvas("rf208_convolution","rf208_convolution",600,600) ;
  gPad->SetLeftMargin(0.15) ; frame->GetYaxis()->SetTitleOffset(1.4) ; frame->Draw() ;

}
```

可以把ROOT作为普通库函数进行连接

```
wsg@debian:/media/sf_UbuntuShare/rootEdu/CmakeTest/cmakeRunCode/GuiRunCode$ ll
total 12
-rwxrwx--- 1 root vboxsf 1834 Aug 12 15:23 CMakeLists.txt
-rwxrwx--- 1 root vboxsf 9645 Jul  8 18:51 FindROOT.cmake
drwxrwx--- 1 root vboxsf    0 Jul  7 23:15 include
-rwxrwx--- 1 root vboxsf  411 Aug 12 15:25 runTest.C
drwxrwx--- 1 root vboxsf    0 Jul  7 23:09 src
```

runTest.C

```
#include "Tools.hh"
#include <TCanvas.h>
#include <TApplication.h>
#include <TRint.h>
int main(int argc, char *argv[]){
  TApplication *theApp = new TRint("App", &argc, argv);

  SetSgStyle();
  printf("run...\n");
  TH1D *h = newTH1D("h",0.1,-5,5);
  h->FillRandom("gaus",10000);
  TCanvas *c1 = new TCanvas("c1","");
  h->Draw();
  txtM(0.2,0.94,h);
  c1->SaveAs("test.eps");

  theApp->Run();
  return 0;
}
```

FindROOT.cmake (http://root.cern.ch/drupal/sites/default/files/event.tgz)
直接放在主目录下

参照CMakeLists.txt建立自己的文件

Include目录放头文件(Tools.hh)
Src目录放.cc文件（Tools.cc）

可以把ROOT作为普通库函数进行连接

CMakeLists.txt

```
#----------------------------------------------------------------------------
# Setup the project
#
cmake_minimum_required(VERSION 2.6 FATAL_ERROR)
project(runTest)



#----------------------------------------------------------------------------
# Find ROOT (Modified it as an option in future)
#set(CMAKE_MODULE_PATH $ENV{G4MODULES} ${CMAKE_MODULE_PATH})
#set(CMAKE_MODULE_PATH $ENV{ROOTSYS}/cmake/modules/  ${CMAKE_MODULE_PATH})
set(CMAKE_MODULE_PATH ${CMAKE_SOURCE_DIR}  ${CMAKE_MODULE_PATH})

find_package(ROOT)
if(ROOT_FOUND)
  message(STATUS "ROOT found.")
else()
    message(STATUS "ROOT not found")
endif()


#----------------------------------------------------------------------------

include_directories(${PROJECT_SOURCE_DIR}/include
                    ${ROOT_INCLUDE_DIR} )
```

```
#----------------------------------------------------------------
# Locate sources and headers for this project
# NB: headers are included so they will show up in IDEs
#
file(GLOB sources ${PROJECT_SOURCE_DIR}/src/*.cc
          )
file(GLOB headers ${PROJECT_SOURCE_DIR}/include/*.hh
          )

#----------------------------------------------------------------
# Add the executable, and link it to the Geant4 libraries
#
add_executable(runTest runTest.C ${sources} ${headers})
target_link_libraries(runTest  ${ROOT_LIBRARIES} )
```

可以把ROOT作为普通库函数进行连接

编译及运行

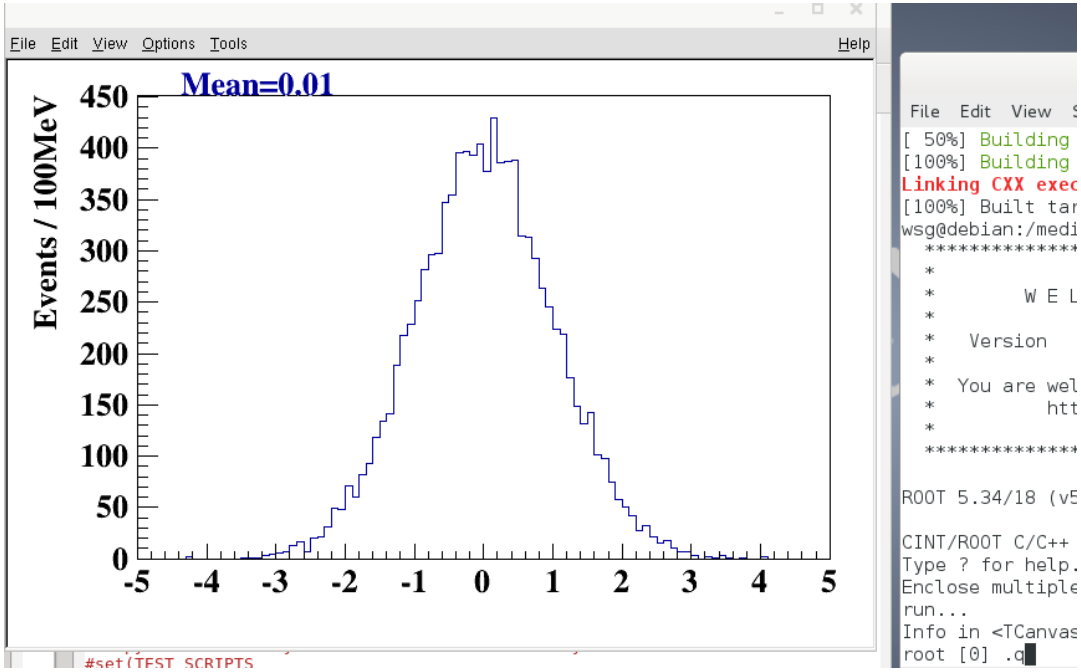
```
>cmake ../GuiRunCode/
>make
>./runTest
root[0].q
```

注：后者为CmakeLists.txt 所在的目录

注：生成可执行程序 runTest

注：运行

注： .q 退出



Siguang@pku.edu.cn

结束语

今天仅仅汇报了我所了解的ROOT部分功能，希望能对您有一些可取之处。

谢谢！

Siguang@pku.edu.cn

附:

Tools.cc 的部分函数

```
void SetSgStyle(){
  // No Canvas Border
  gStyle->SetCanvasBorderMode(0);
  gStyle->SetCanvasBorderSize(0);
  // White BG
  gStyle->SetCanvasColor(10);
  // Format for axes
  gStyle->SetLabelFont(22,"xyz");
  gStyle->SetLabelSize(0.06,"xyz");
  gStyle->SetLabelOffset(0.01,"xyz");
  gStyle->SetNdivisions(510,"xyz");
  gStyle->SetTitleFont(22,"xyz");
  gStyle->SetTitleColor(1,"xyz");
  gStyle->SetTitleSize(0.06,"xyz");
  gStyle->SetTitleOffset(0.91);
  gStyle->SetTitleYOffset(1.1);
  // No pad borders
  gStyle->SetPadBorderMode(0);
  gStyle->SetPadBorderSize(0);
  // White BG
  gStyle->SetPadColor(10);
  // Margins for labels etc.
  gStyle->SetPadLeftMargin(0.15);
  gStyle->SetPadBottomMargin(0.15);
  gStyle->SetPadRightMargin(0.05);
  gStyle->SetPadTopMargin(0.06);
  // No error bars in x direction
  gStyle->SetErrorX(0);

  // Format legend
  gStyle->SetLegendBorderSize(0);
  gStyle->SetLegendFont(22);
  gStyle->SetFillStyle(0);
}
```

```
TH1D * newTH1D(TString name,Double_t binw, Double_t LowBin, Double_t Hig
hBin,Bool_t MevTitle,Int_t iMode){
  Int_t nbin = TMath::Nint( (HighBin - LowBin)/binw );
  HighBin = binw*nbin + LowBin;

  TH1D *h = new TH1D(name.Data(),"",nbin,LowBin,HighBin);
  if(MevTitle) h->GetYaxis()->SetTitle(Form("Events / %.0fMeV",h->GetBin
Width(1)*1000));
  h->SetMinimum(0.0);
  h->GetYaxis()->SetTitleOffset(1.1);
  if(iMode>=0 && iMode<14){
    Int_t iMarker[] = {20,21,24,25,28,29,30,27,3,  5,2, 26,22,23};
    Int_t iColor[]  = { 2, 4, 6, 9, 1,50,40,31,41,35,44,38,47,12};
    h ->SetMarkerStyle(iMarker[iMode]);
    h ->SetMarkerColor(iColor[iMode]);
    h ->SetLineColor(iColor[iMode]);
  }

  return h;
}
```