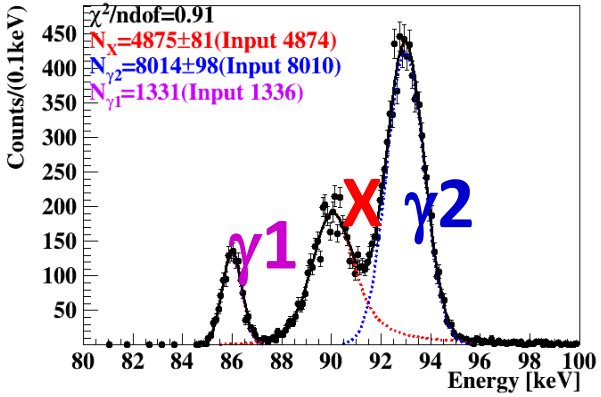
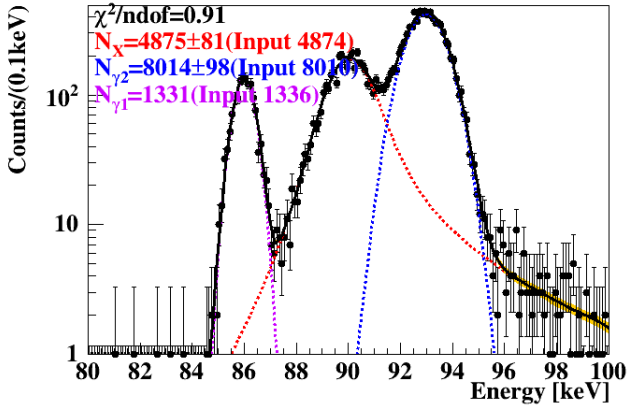


ROOT介绍



王思广
Email: siguang@pku.edu.cn

北京大学物理学院

The 3rd International Summer School on TeV Experimental Physics
Tsinghua University, Jul. 13, 2016

讲课内容

自学顺序

① ROOT介绍、安装及运行检查 P.3

② 利用RooFit进行重叠峰拟合 P.28

③ ROOT 部分功能展示 P.43

④ Codes under $ROOTSYS/tutorials/

① Graphics P.126

② Graphs P.202

③ Hist P.252

⑤ A ROOT Guide For Beginners P.340

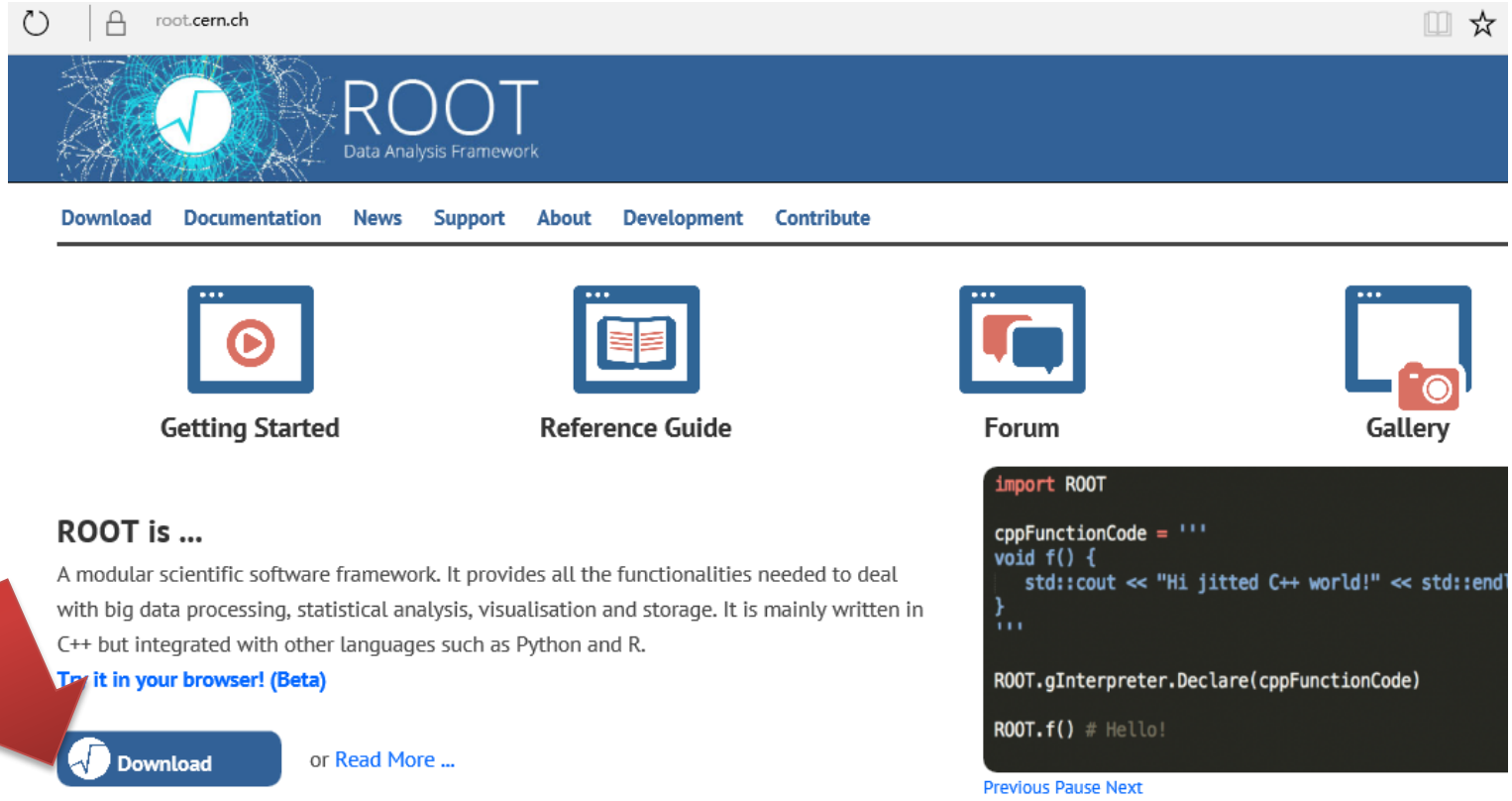
TMVA - Toolkit for Multivariate Data Analysis P.404

本报告下载地址：
http://www.phy.pku.edu.cn/~wangsg/ROOTSchool/sgROOTIntroTeV2016.pdf

ROOT介绍、安装及运行检查

ROOT自由软件网页

https://root.cern.ch/



siguang@pku.edu.cn

ROOT V5

root.cern.ch/downloading-root

Pro Release 6.06/02 - 2016-03-03

Old Release 6.04/16 - 2016-03-17

Version 6

Release 6.06/04 - 2016-05-03

Release 6.04/16 - 2016-03-17

Release 6.06/02 - 2016-03-03

Release 6.04/14 - 2016-02-03

Release 6.06/00 - 2015-12-09

Release 6.04/12 - 2015-12-04

Release 6.04/10 - 2015-11-18

Release 6.04/08 - 2015-11-04

Release 6.04/06 - 2015-10-13

Release 6.04/04 - 2015-10-08

Release 6.05/02 - 2015-09-14

Release 6.04/02 - 2015-07-14

Release 6.02/12 - 2015-06-24

Release 6.04/00 - 2015-06-02

Release 6.02/10 - 2015-05-29

Version 5

Release 5.34/36 - 2016-04-05

Release 5.34/34 - 2015-10-02

Release 5.34/32 - 2015-06-23

See a full list of the releases here.

ROOT V5

Source distribution

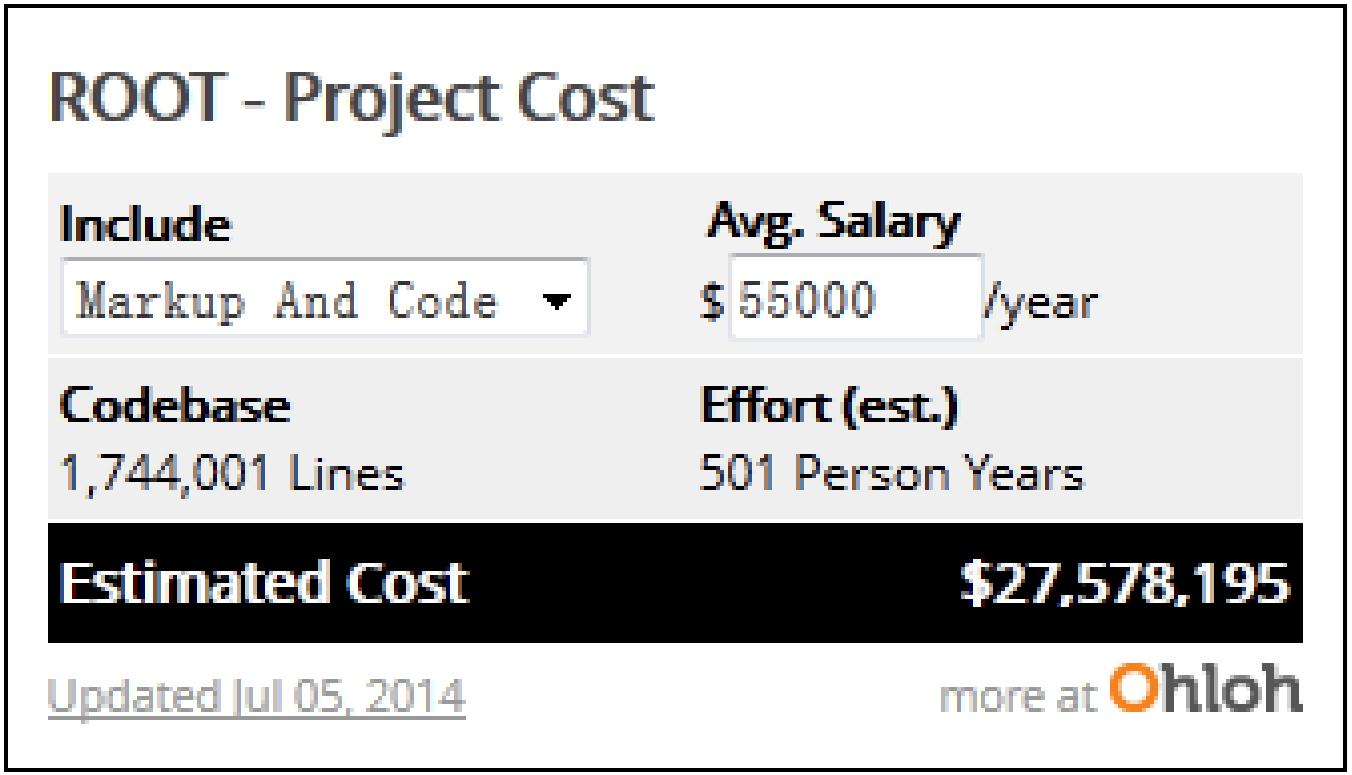
Platform	Files	Size
source	root_v5.34.36.source.tar.gz	72M

Binary distributions

Platform	Files	Size
CentOS Cern 7 gcc4.8	root_v5.34.36.Linux-centos7-x86_64-gcc4.8.tar.gz	72M
CentOS Cern 7 gcc4.9	root_v5.34.36.Linux-centos7-x86_64-gcc4.9.tar.gz	73M
Linux fedora20 gcc4.8	root_v5.34.36.Linux-fedora20-x86_64-gcc4.8.tar.gz	58M
Scientific Linux Cern 6 gcc4.4	root_v5.34.36.Linux-slc6-x86_64-gcc4.4.tar.gz	70M
Scientific Linux Cern 6 gcc4.7	root_v5.34.36.Linux-slc6-x86_64-gcc4.7.tar.gz	71M
Scientific Linux Cern 6 gcc4.8	root_v5.34.36.Linux-slc6-x86_64-gcc4.8.tar.gz	71M
Scientific Linux Cern 6 gcc4.9	root_v5.34.36.Linux-slc6-x86_64-gcc4.9.tar.gz	73M
Scientific Linux Cern 6 gcc5.1	root_v5.34.36.Linux-slc6-x86_64-gcc5.1.tar.gz	73M
Ubuntu 12 gcc4.6	root_v5.34.36.Linux-ubuntu12-x86_64-gcc4.6.tar.gz	58M
Ubuntu 14 gcc4.8	root_v5.34.36.Linux-ubuntu14-x86_64-gcc4.8.tar.gz	62M
OsX 10.9 clang60	root_v5.34.36.macosx64-10.9-clang60.dmg	56M
OsX 10.9 clang60	root_v5.34.36.macosx64-10.9-clang60.tar.gz	56M

Project Statistics

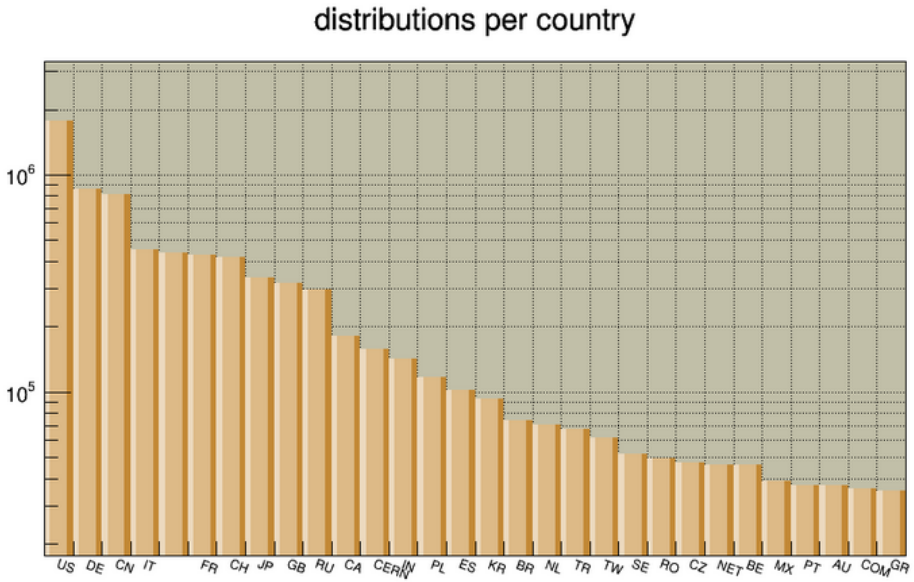
http://root.cern.ch/drupal/content/project-statistics



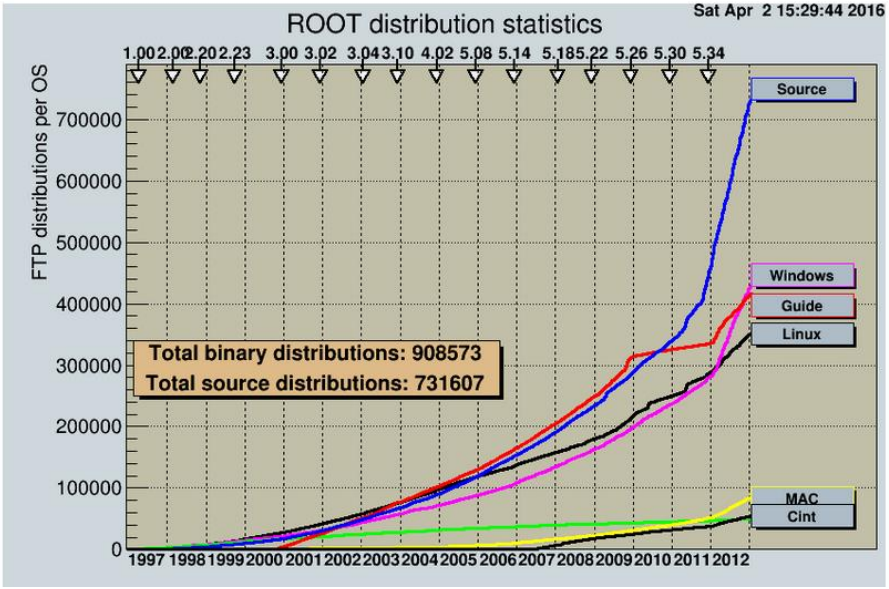
2014年统计的，后来就没找到类似网页

各国下载量

ROOT Total Download Statistics per country



ROOT Binary Distribution Statistics



https://root.cern.ch/drupal/content/download-statistics

2016.4.2有效

在Windows下安装虚拟机方法

1) 先安装VirtualBox
https://virtualbox.org/

2)导入安装好的文件或自己安装Linux系统然后安装root

王思广所提供的虚拟机安装方式

所提供的虚拟机引导Debian操作系统，VirtualBox版本：

VirtualBox-4.3.12-93733-Win (其它版本的VirtualBox发现无法使用）

第一步：下载地址（64位）：

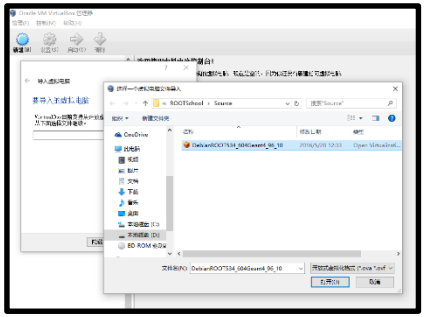
http://www.phy.pku.edu.cn/~wangsg/ROOTSchool/VMVirtualBox_Debian_ROOT_G4.rar

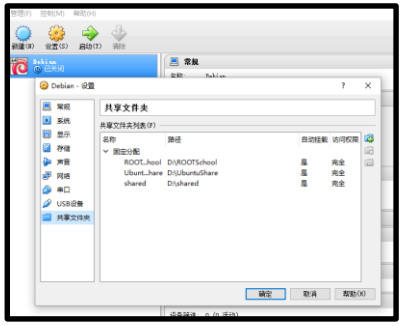
下载压缩文件并解压,找到VirtualBox-4.3.12-93733-Win.exe并安装;

第二步：在“管理”下拉菜单“导入虚拟电脑”弹出的选择框内选择“DebianROOT534_604Geant4_96_10.ova”

第三步：按“设置”后进行“共享文件夹”设置，Window与虚拟机之间可以通过共享文件夹进行数据交换

第四步：“启动”，对于wsg用户和超级root用户，密码都是testroot





32位下载：

http://www.phy.pku.edu.cn/~wangsg/ROOTSchool/VirtualBox-4.3.12-93733-Win.exe

http://www.phy.pku.edu.cn/~wangsg/ROOTSchool/VBoxGuestAdditions_4.3.12.iso

http://www.phy.pku.edu.cn/~wangsg/ROOTSchool/Debin32bit_ROOT534_600_G496p02.ova

Configure安装ROOT方法：含fftw Pythia6 及Pythia8扩展包

```
#!/bin/bash
export PYTHIA6=/home/wsg/work/pythia6/pythia6428
export PYTHIA8=/home/wsg/work/pythia8/8186
export PYTHIA8DATA=/home/wsg/work/pythia8/8186/xmldoc
export PYTHONDIR=/usr
export PATH=$PYTHONDIR/bin:$PATH
export LD_LIBRARY_PATH=$PYTHONDIR/lib:$PYTHIA6:$PYTHIA8/lib:$LD_LIBRARY_PATH
export PYTHONPATH=$PYTHONDIR/lib:$PYTHONPATH

./configure --prefix=/home/wsg/work/root/534 --fail-on-missing --enable-pythia8  --with-pythia8-
incdir=$PYTHIA8/include  --with-pythia8-libdir=$PYTHIA8/lib --enable-pythia6 --with-pythia6-
libdir=$PYTHIA6  --enable-fftw3 --with-fftw3-incdir=/usr/include --with-fftw3-libdir=/usr/lib --enable-
python  --with-python-incdir=/usr/include/python2.7 --with-python-libdir=/usr/lib  --enable-tmva --
enable-qt --enable-unuran --enable-qtgsi  --enable-minuit2 --enable-roofit  --enable-gdml  --enable-
reflex  --enable-cxx11 --enable-cocoa
make -j2
```

安装前要进行PYTHIA fftw的包的安装

FFTW的安装

方法1：Debian, Ubuntu等系统源自带的安装方法（发现这种方法更安全稳定，推荐！）：

apt-get install libfftw3-dev

方法2：下载FFTW源代码(网站见下页)，解压后查看安装说明

emacs INSTALL　　看看安装方法

mkdir fftw　　安装目录随便建立:

/home/wsg/work/fftw

./configure - -prefix=/home/wsg/work/fftw

【prefix前是两个减号】

make –j2　这里的2指CUP数目，可并行编译

make install

http://www.fftw.org/download.html

Downloading FFTW

如果需要RooFit的快速傅里叶卷积拟合，
需要在编译安装root前安装FFTW

Mailing list / Announcements

Subscribe to the fftw-announce mailing list on Google Groups to receive an email when FFTW is updated in the future. Alternatively, you can use the web feed or mailing list on freecode.com.

You can contact the FFTW authors at fftw@fftw.org.

FFTW 3.3.3

Version 3.3.3 is the latest stable release of FFTW, and full source code is found here:

- **http:** fftw-3.3.3.tar.gz (**ftp:** fftw-3.3.3.tar.gz) (md5sum) (4.0MB)

PYTHIA6.4.28 安装方法（需要部分更新代码）

http://home.thep.lu.se/~torbjorn/Pythia.html

```
wget https://root.cern.ch/download/pythia6.tar.gz
tar zxvf pythia6.tar.gz
rm -rf pythia6.tar.gz
wget http://www.hepforge.org/archive/pythia6/pythia-6.4.28.f.gz
gzip -d pythia-6.4.28.f.gz
mv pythia-6.4.28.f pythia6/pythia6428.f
rm -rf pythia6/pythia6416.f
mv pythia6 pythia6428
cd pythia6428
./makePythia6.linuxx8664
cd ..

wget http://home.thep.lu.se/~torbjorn/pythia8/pythia8186.tgz
tar zxvf pythia8186.tgz
rm -rf pythia8186.tgz
cd pythia8186
./configure --enable-shared --enable-64bit
make -j2
cd ..
```

建立环境变量设置脚本

```
cat >setupPythiaROOT

#!/bin/sh
export PYTHIA6=$PWD/pythia6428
export PYTHIA8=$PWD/pythia8186
export PYTHIA8DATA=$PWD/pythia8186/xmldoc

source setupPythiaROOT
```

安装过程中可能遇到困难怎么办？

更新系统：
sudo apt-get update 更新源
sudo apt-get upgrade 更新已安装的包

如果还出现问题，仔细看错误提示，安装相应的软件包。如果安装ROOT6，需要gcc4.8的版本，Debian下更新gcc的方法：

sudo apt-get update 更新源
sudo apt-get upgrade 更新已安装的包
sudo cp /etc/apt/sources.list /etc/apt/sources.list.WHEEZY 备份
emacs /etc/apt/sources.list 编辑，替换所有的“wheezy”为“jessie”
sudo apt-get update更新源
sudo apt-get install gcc-4.9 g++-4.9 安装
sudo cp /etc/apt/sources.list.WHEEZY /etc/apt/sources.list
sudo apt-get update

ROOT 需要的包

根据
https://root.cern.ch/drupal/content/build-prerequisites
需要如下的包：

不同操作系统所需要的包见该网页

sudo apt-get install git dpkg-dev make g++ gcc binutils libx11-dev libxpm-dev libxft-dev libxext-dev

Optional packages:

sudo apt-get install gfortran libssl-dev libpcre3-dev xlibmesa-glu-dev libglew1.5-dev libftgl-dev libmysqlclient-dev libfftw3-dev cfitsio-dev graphviz-dev libavahi-compat-libdnssd-dev libldap2-dev python-dev libxml2-dev libkrb5-dev libgsl0-dev libqt4-dev

如果有ROOT权限，执行以上命令即可

apt-get命令

apt-get近乎是最常用的shell命令之一了，
常用的APT命令参数：
apt-cache search package 搜索软件包
apt-cache show package 获取包的相关信息，如说明、大小、版本等
sudo apt-get install package 安装包
sudo apt-get install package --reinstall 重新安装包
sudo apt-get -f install 修复安装
sudo apt-get remove package 删除包
sudo apt-get remove package --purge 删除包，包括配置文件等
sudo apt-get update 更新源
sudo apt-get upgrade 更新已安装的包
sudo apt-get dist-upgrade 升级系统
apt-cache depends package 了解使用该包依赖那些包
apt-cache rdepends package 查看该包被哪些包依赖
sudo apt-get build-dep package 安装相关的编译环境
apt-get source package 下载该包的源代码
sudo apt-get clean && sudo apt-get autoclean 清理无用的包
sudo apt-get check 检查是否有损坏的依赖

将gv安装：
apt-get install gv

也可安装操作系统源的root

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

tutorials、test目录会被安装在：

/usr/share/doc/root/tutorials

和

/usr/share/doc/root/test

Linux下通过cmake编译源代码安装

前期准备：

- cmake；
- 下载root源代码解压缩到root目录下

```
mkdir -p tmpRootCompile
cd tmpRootCompile
cmake ../root-6.04.16 -DCMAKE_INSTALL_PREFIX=/home/wsg/work/root/604
-Dall=on  -Dfail-on-missing=OFF

make -j2
make install
```

root是源代码解压后的目录
/home/wsg/work/root/604是要安装 后的目录
-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
详细见： https://root.cern.ch/installing-root-source

Linux下通过cmake编译源代码安装root5

安装脚本：

```
#!/bin/bash
#pre install FFTW with apt-get install libfftw3-dev
#more: https://root.cern.ch/installing-root-source
export PYTHIA6=/home/wsg/work/pythia6/pythia6428
export PYTHIA8=/home/wsg/work/pythia8/8186
export PYTHIA8DATA=/home/wsg/work/pythia8/8186/xmldoc
export PYTHONDIR=/usr
export PYTHONPATH=$PYTHONDIR/lib
export PATH=$PYTHONDIR/bin:$PATH
export LD_LIBRARY_PATH=$PYTHONDIR/lib:$PYTHIA6:$PYTHIA8/lib:$PYTHONDIR/lib:$LD_LIBRARY_PATH

mkdir -p tmpRootCompile
cd tmpRootCompile
```

Root534下只能安装pythia8.1*
Root6下可以安装pythia8.2*
设置pythia8*环境的时候：
./configure --enable-shared --enable-64bit –prefix=……..

cmake 安装细节

cmake -DCMAKE_INSTALL_PREFIX=/home/wsg/work/root/534
../root
-DPYTHIA6_LIBRARY=/home/wsg/work/pythia6/pythia6428/libPythia6.so -Dpythia6=ON
-DPYTHIA8_DIR=/home/wsg/work/pythia8/8186
-DPYTHIA8_INCLUDE_DIR=/home/wsg/work/pythia8/8186/include
-DPYTHIA8_LIBRARY=/home/wsg/work/pythia8/8186/lib/libpythia8.so -Dpythia8=on
-DPYTHON_EXECUTABLE=/usr/bin
-DPYTHON_INCLUDE_DIR=/usr/include/python2.7
-DPYTHON_INCLUDE_DIR2=/usr/include/python2.7
-DPYTHON_LIBRARY=/usr/lib/python2.7/config/libpython2.7.so
-Dall=on -Droofit=on -Dfftw3=on -Dpython=on -Droottest=on -Druby=on -Dtmva=on -Dtestring=on -Dxml=on -Dx11=on -Dqt=on -Dmt=on -Dxrootd=on -Dtcmalloc=on -Dfail-on-missing=OFF

以上真实代码在一行

make –j2

make install

root是源代码解压后的目录
/home/wsg/work/root/534是要安装 后的目录
-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
详细见： https://root.cern.ch/installing-root-source

设置环境

检查运行环境： echo $0

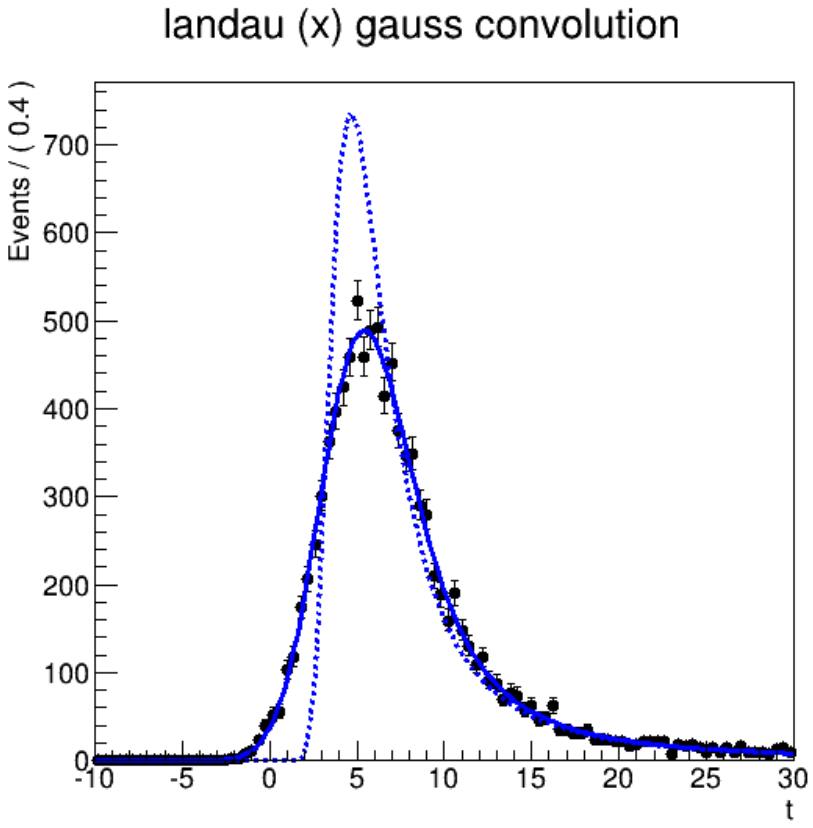
如果返回bash
wsg@debian:~$ cd
wsg@debian:~$ emacs .bashrc &
在文件中加入：
export PYTHONPATH=/usr/lib/python2.7　为了使得pyroot可用
export ROOTSYS=/home/wsg/work/root/534
export PATH=$ROOTSYS/bin:$PATH
export LD_LIBRARY_PATH=$ROOTSYS/lib:$LD_LIBRARY_PATH

如果返回-tcsh
wsg@debian:~$ cd
wsg@debian:~$ emacs .tcshrc &
在文件中加入：
setenv PYTHONPATH /usr/lib/python2.7　　为了使得pyroot可用
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

安装后的运行测试： $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。

siguang@pku.edu.cn

编译运行方法

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()
```

$ROOTSYS/tutorials/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) ;
```


siguang@pku.edu.cn

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() ;

}
```

利用RooFit进行重叠峰拟合

TRandom3

周期10^{6000}

```
public:
                TRandom3 (UInt_t seed = 4357)
                TRandom3 (const TRandom3&)
        virtual ~TRandom3 ()
   static TClass* Class ()
  virtual TClass* IsA () const
     TRandom3& operator= (const TRandom3&)
virtual Double_t Rndm (Int_t i = 0)
    virtual void RndmArray (Int_t n, Float_t* array)
    virtual void RndmArray (Int_t n, Double_t* array)
    virtual void SetSeed (UInt_t seed = 0)
    virtual void ShowMembers (TMemberInspector& insp, char* parent)
    virtual void Streamer (TBuffer& b)
            void StreamerNVirtual (TBuffer& b)
```

TRandom3 r;
r.SetSeed(0);
Double_t val = r.Rndm();

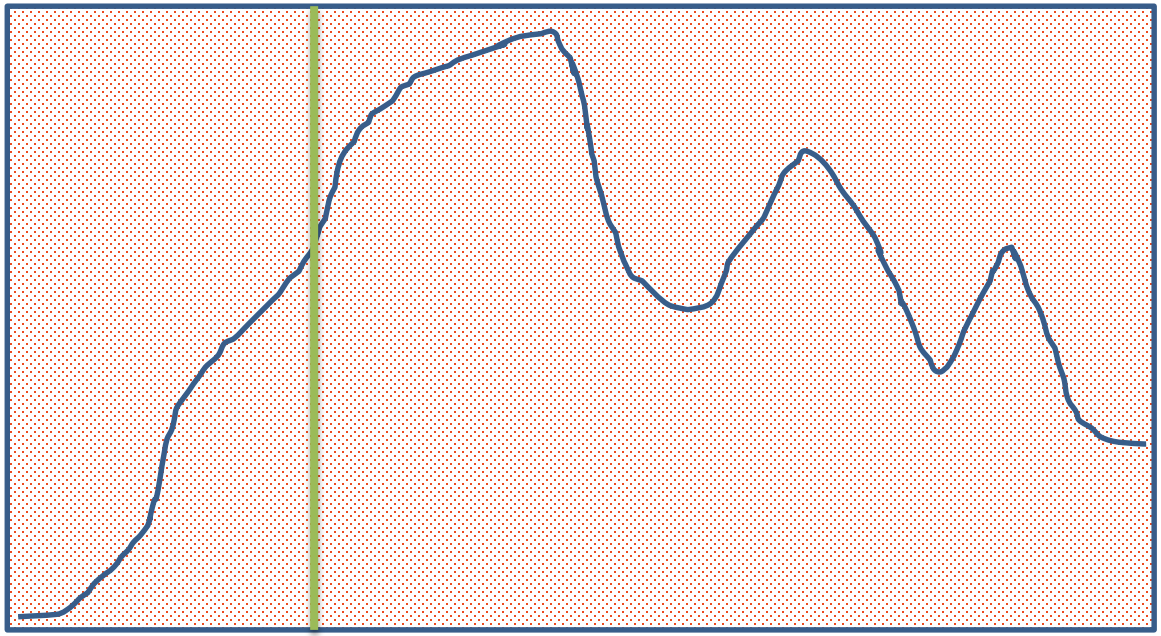
Rndm():
Machine independent random number generator. Produces uniformly-distributed floating points in]0,1]
Method: Mersenne Twistor

SetSeed(0) ---自动随机 or SetSeed(Num), Num>0 固定序列

TRandom3

Distributions	Description
`Double_t Uniform(Double_t x1,Double_t x2)`	Uniform random numbers between `x1,x2`
`Double_t Gaus(Double_t mu,Double_t sigma)`	Gaussian random numbers. Default values: `mu=0, sigma=1`
`Double_t Exp(Double_t tau)`	Exponential random numbers with mean tau.
`Double_t Landau(Double_t mean,Double_t sigma)`	Landau distributed random numbers. Default values: `mean=0, sigma=1`
`Double_t BreitWigner(Double_t mean,` `Double_t gamma)`	Breit-Wigner distributed random numbers. Default values `mean=0, gamma=1`
`Int_t Poisson(Double_t mean)` `Double_t PoissonD(Double_t mean)`	Poisson random numbers
`Int_t Binomial(Int_t ntot,Double_t prob)`	Binomial Random numbers
`Circle(Double_t &x,Double_t &y,Double_t r)`	Generate a random 2D point `(x,y)` in a circle of radius `r`
`Sphere(Double_t &x,Double_t &y,` `Double_t &z,Double_t r)`	Generate a random 3D point `(x,y,z)` in a sphere of radius `r`
`Rannor(Double_t &a,Double_t &b)`	Generate a pair of Gaussian random numbers with `mu=0` and `sigma=1`

一个精确的函数产生方法



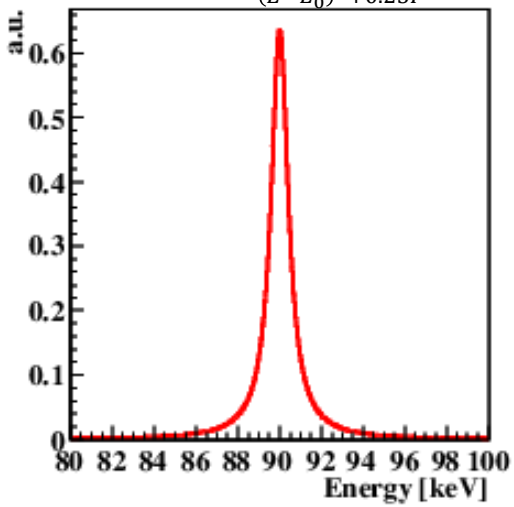
1)随机产生(x,y)坐标
2)判断y是否在线以下

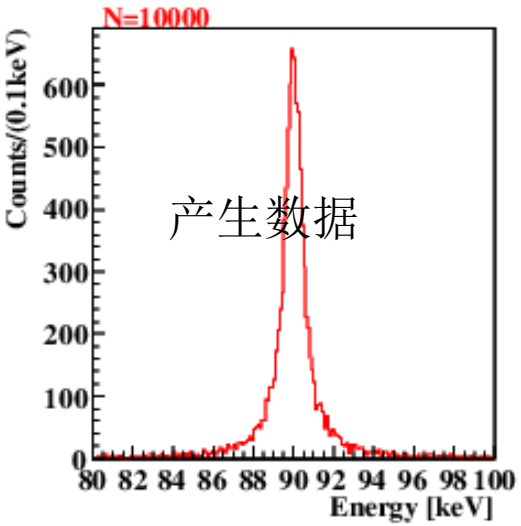
```
TRandom3   r;
TH1F *h=new TH1F("h","；  x；  y，  100,0,1）
r.SetSeed(0);
Double_t X = r.Rndm();
Double_t Y = r.Rndm();
If(Y<f(X) h->Fill(X)
```

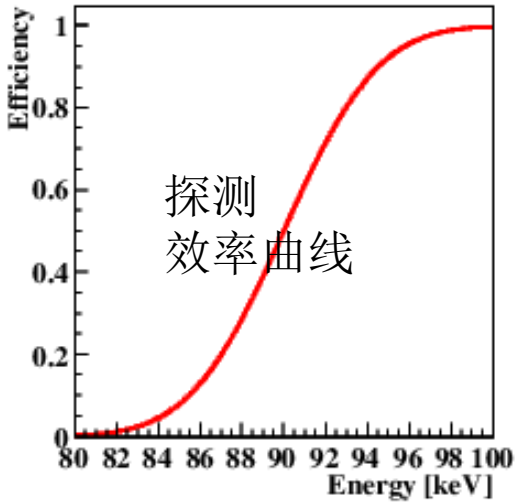

siguang@pku.edu.cn

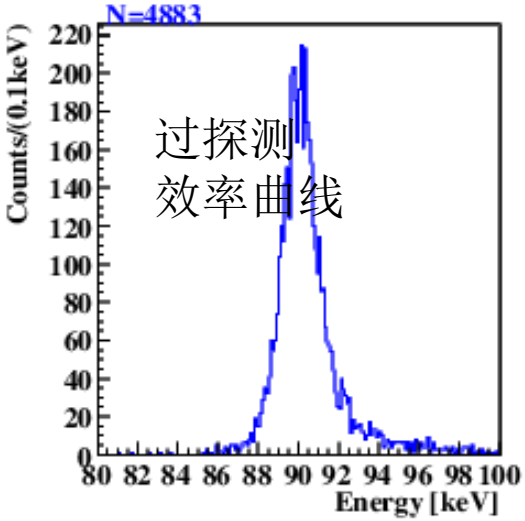
X射线能峰形状

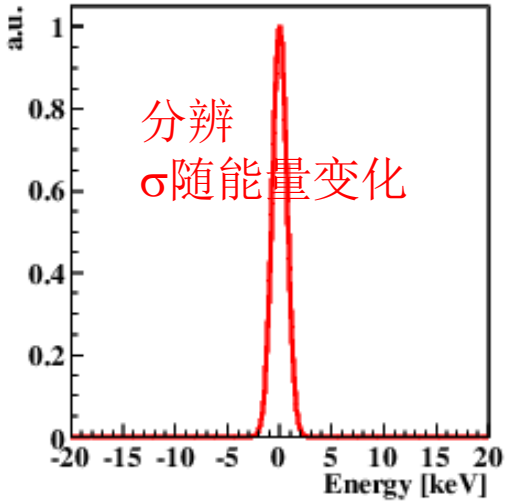
本征形状: $f(E) \propto \frac{1}{(E-E_0)^2+0.25\Gamma^2}$

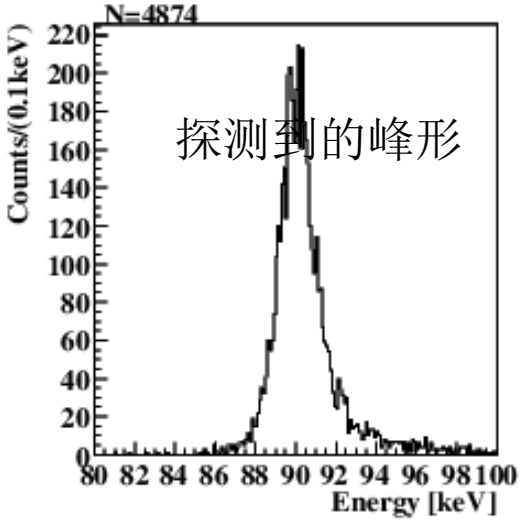






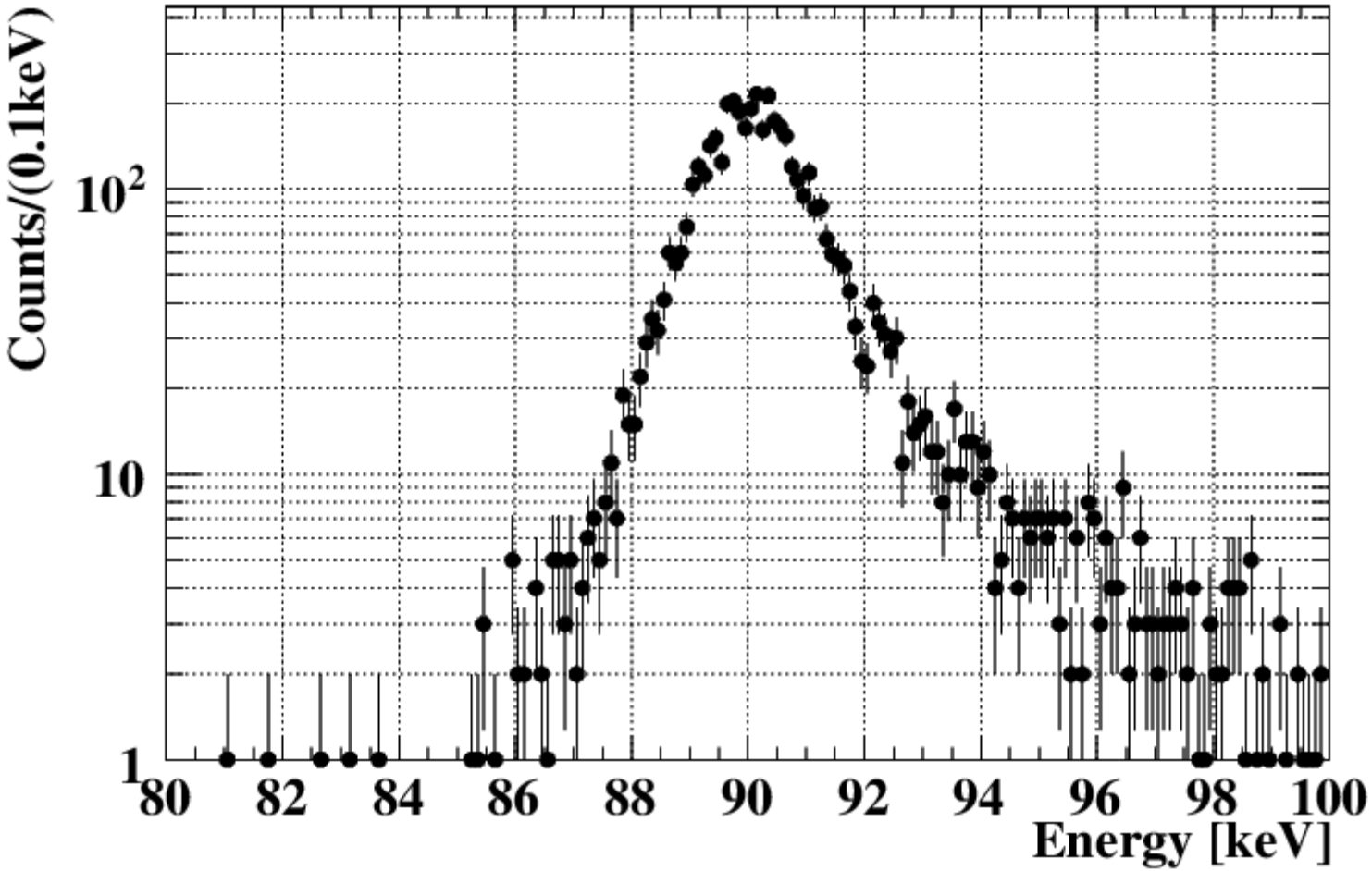






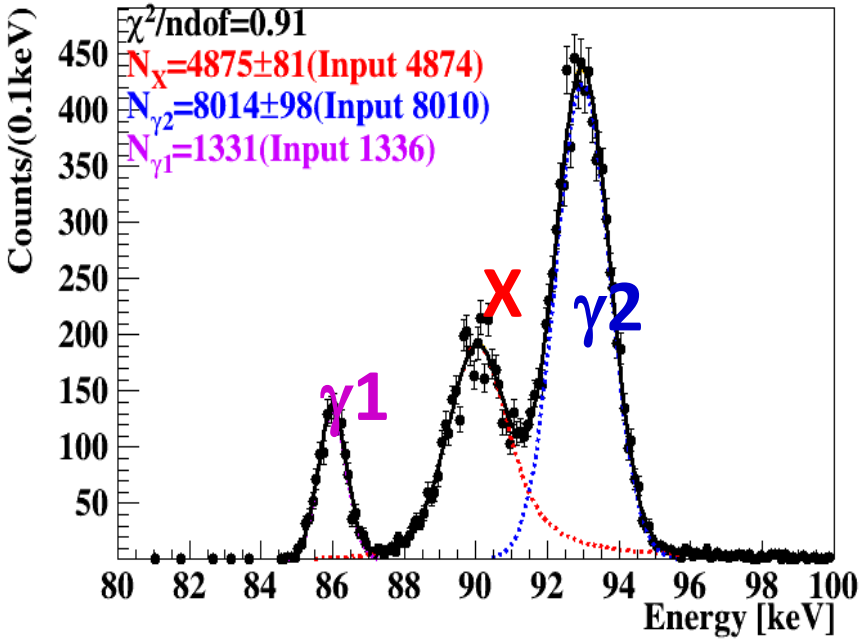
siguang@pku.edu.cn

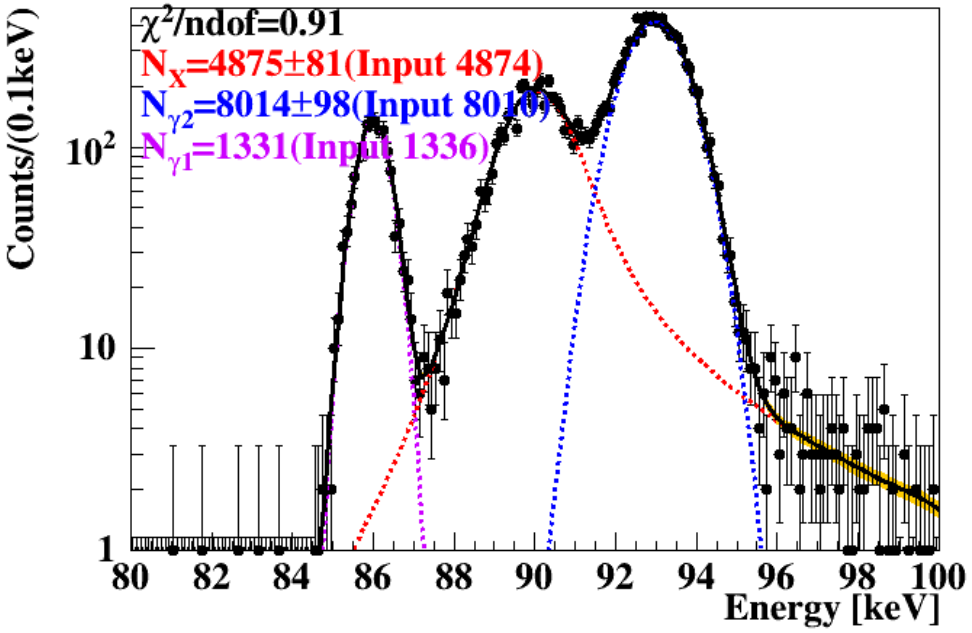
X射线能峰形状(产生)



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.**"）

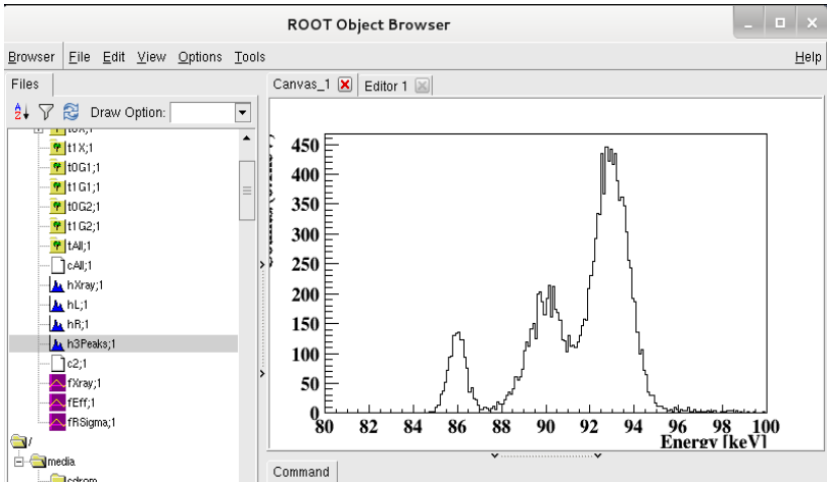
2）分辨是射线能量的函数；对于X射线这里近似用峰位分辨.

3）效率是能量的函数

实现代码 （读入能谱）

```
//
//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();
```

$root spec.root
[] new TBrowser



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(unbin条件下)：

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

siguang@pku.edu.cn

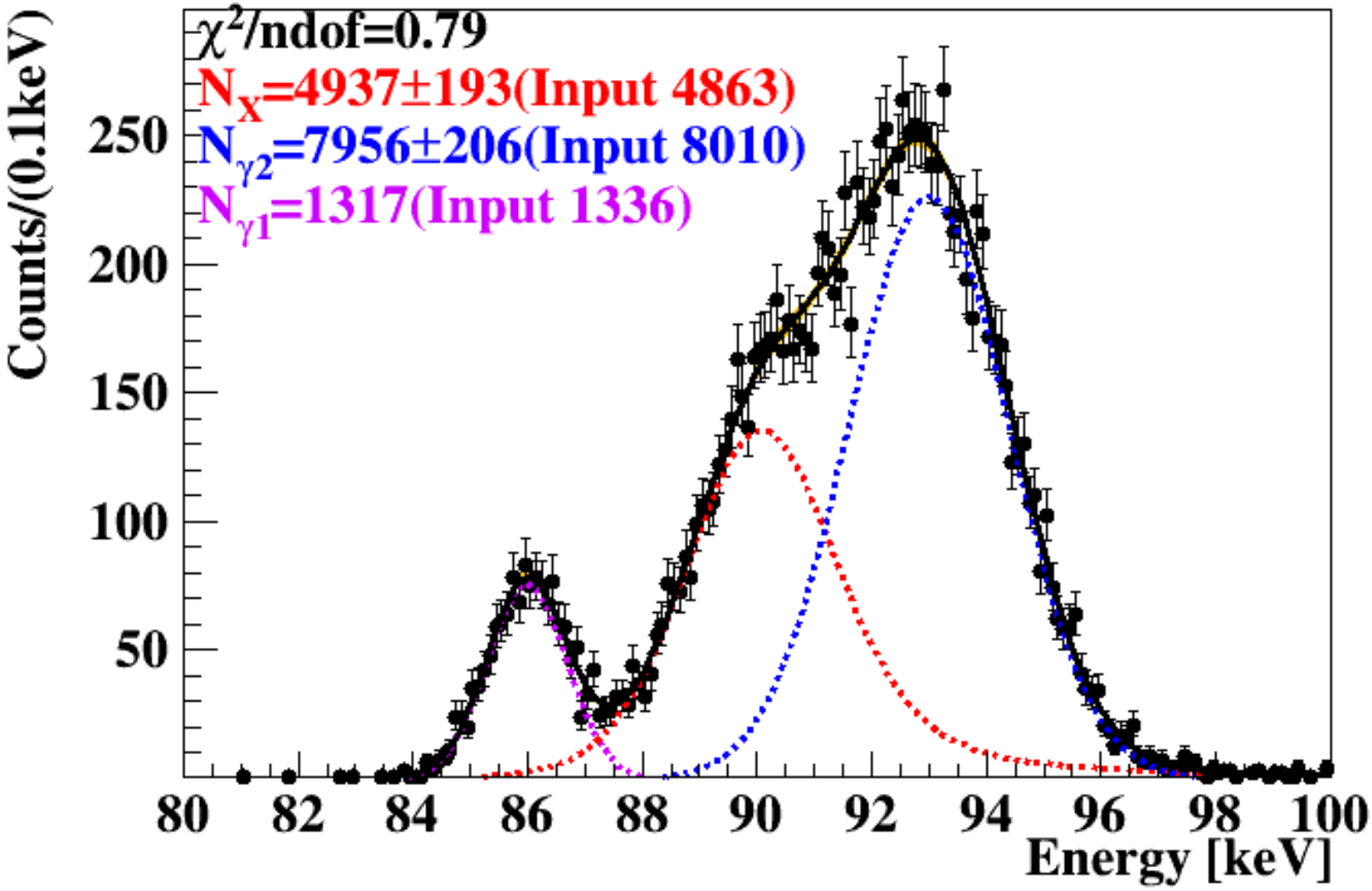
实现代码（作图）

```
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");
```

具体可参考：

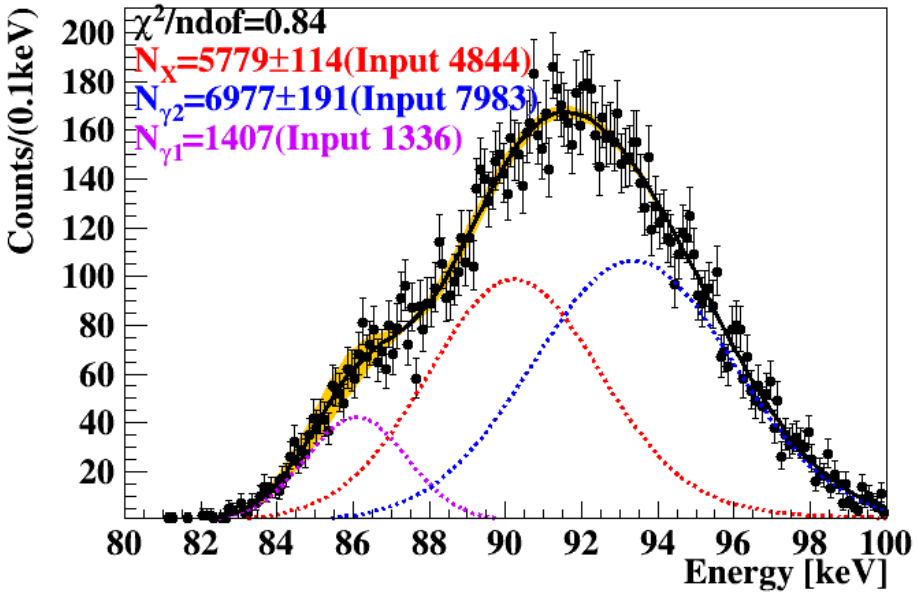
王思广*，韦江波，张欢，X射线峰形描述技术及分支比约束在解谱中的应用，《核技术》 vol **38**, No.2, 020502(2015)

较差分辨下能谱的拟合

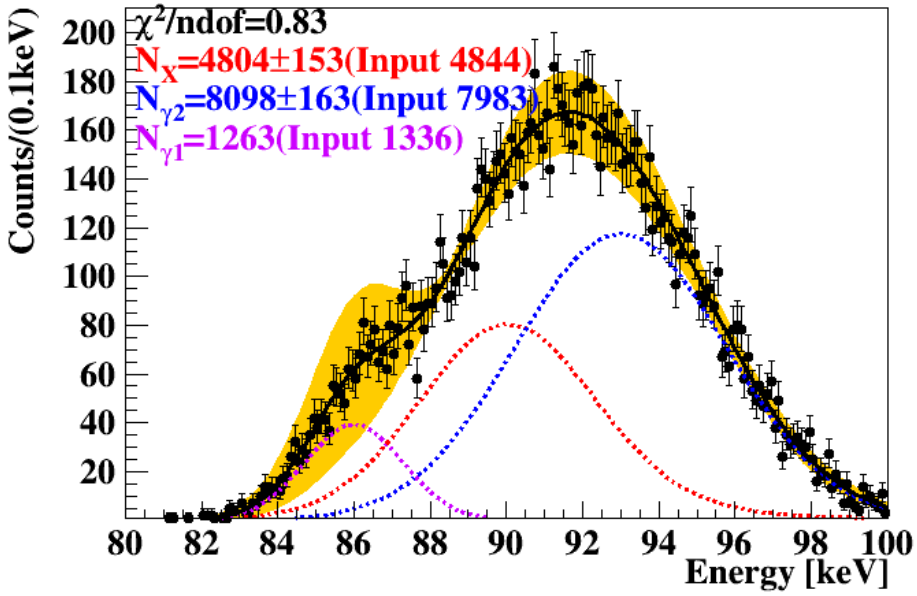


siguang@pku.edu.cn

更差分辨下能谱的拟合



自由浮动峰位

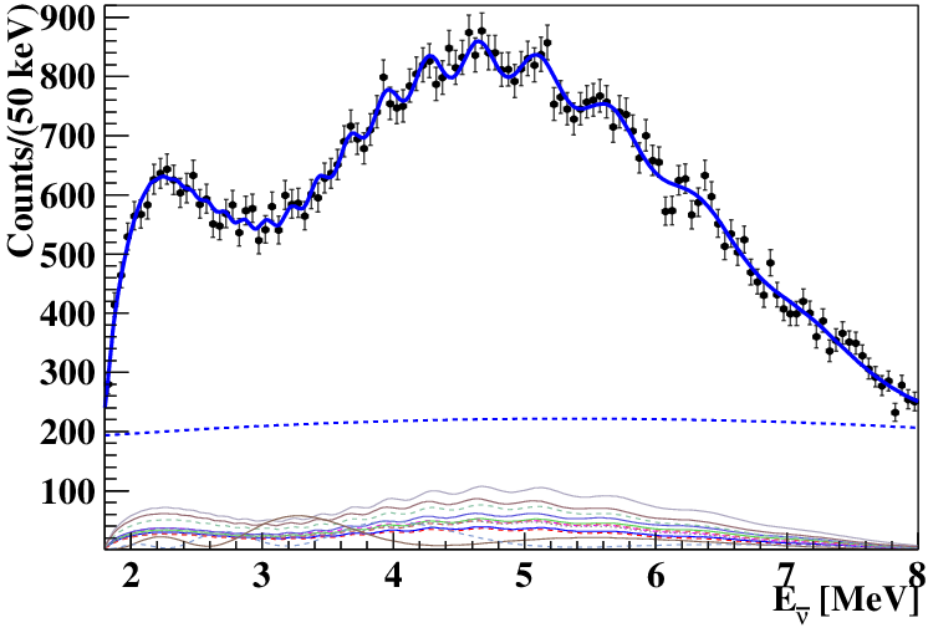


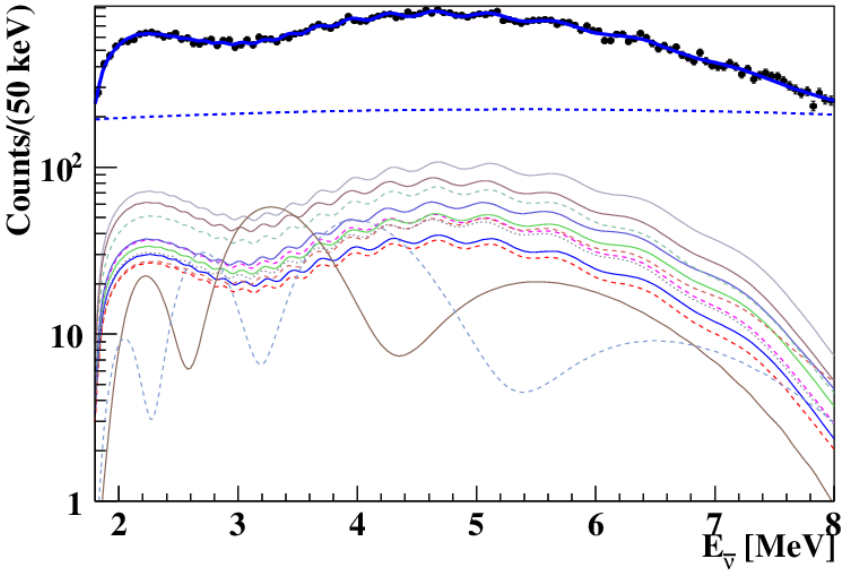
峰位固定

siguang@pku.edu.cn

分辨随能量变化拟合例子

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





55个自由参数的拟合;
需要根据卷积的定于自己写程序

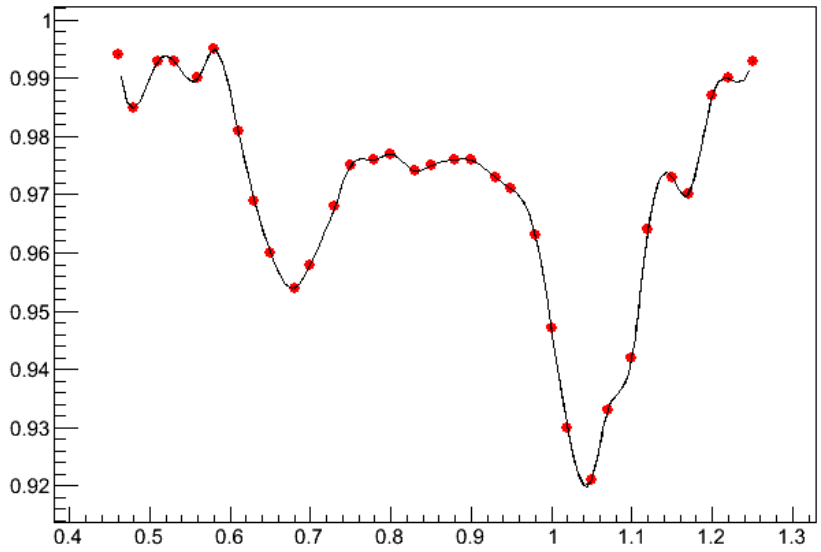
siguang@pku.edu.cn

ROOT 部分功能展示

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));
  }
}
```

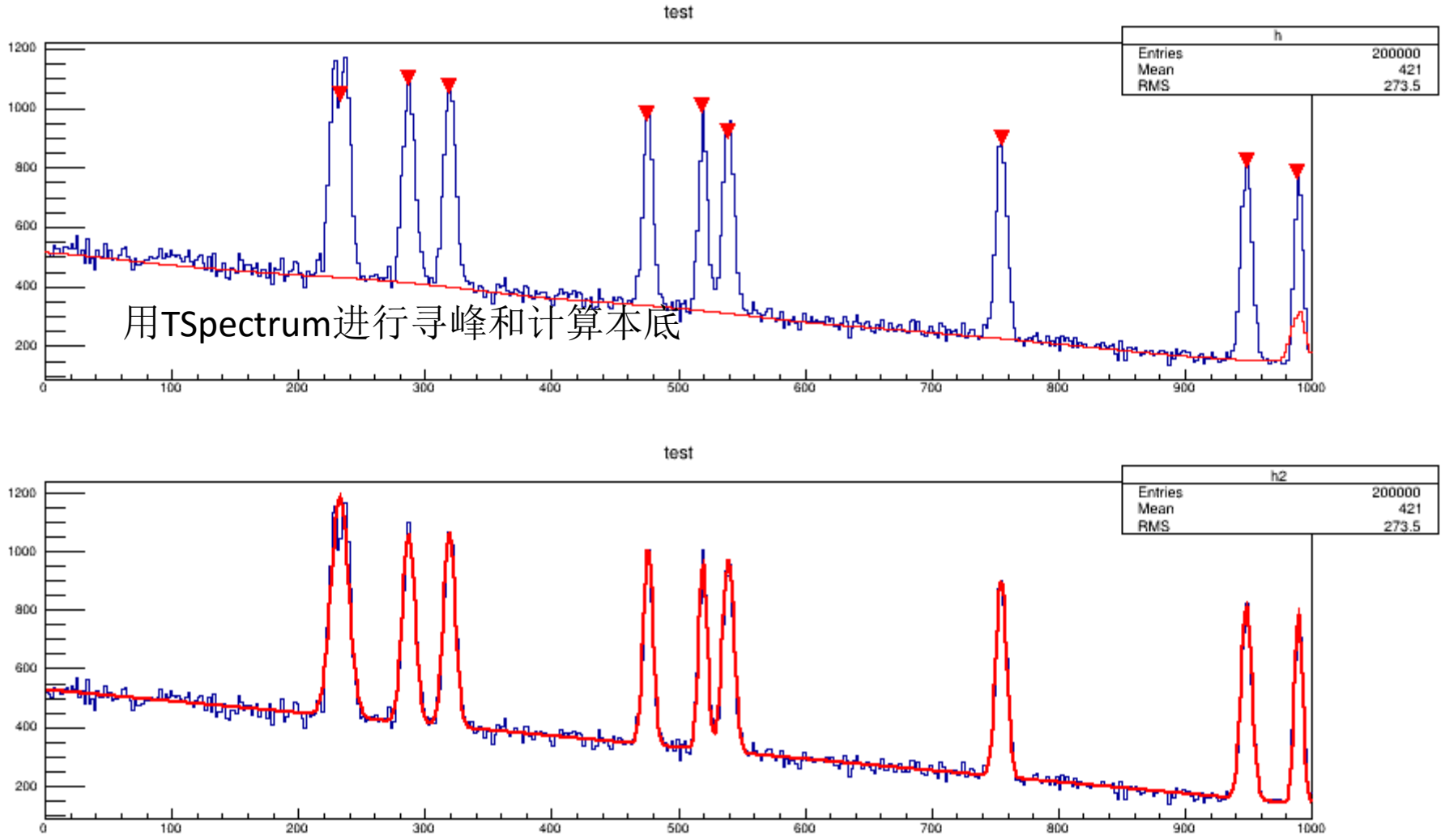
test.txt



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
0.7	0.958
0.73	0.968
0.75	0.975
0.78	0.976
0.8	0.977
0.83	0.974
0.85	0.975
0.88	0.976
0.9	0.976
0.93	0.973
0.95	0.971
0.98	0.963
1	0.947
1.02	0.93

siguang@pku.edu.cn

ROOT自带的能谱分析类TSpectrum

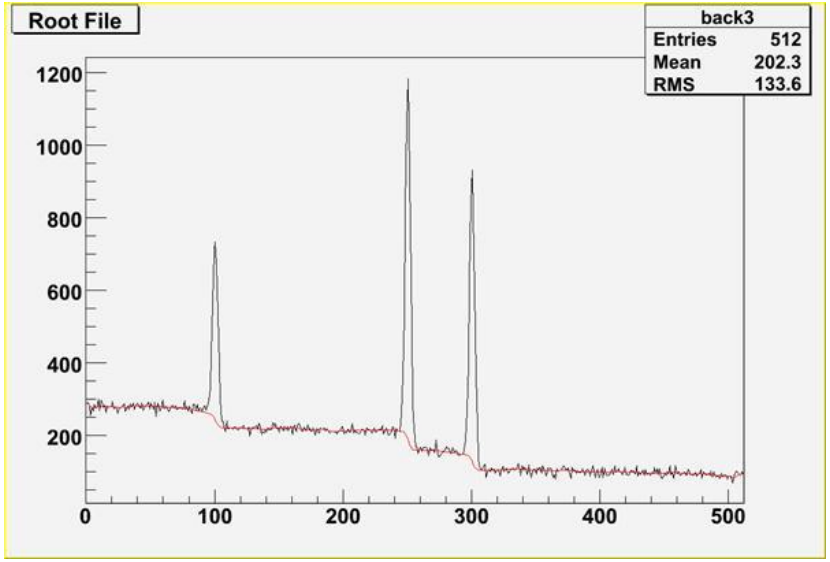


$ROOTSYS/tutorials/spectrum $ root peaks.C

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

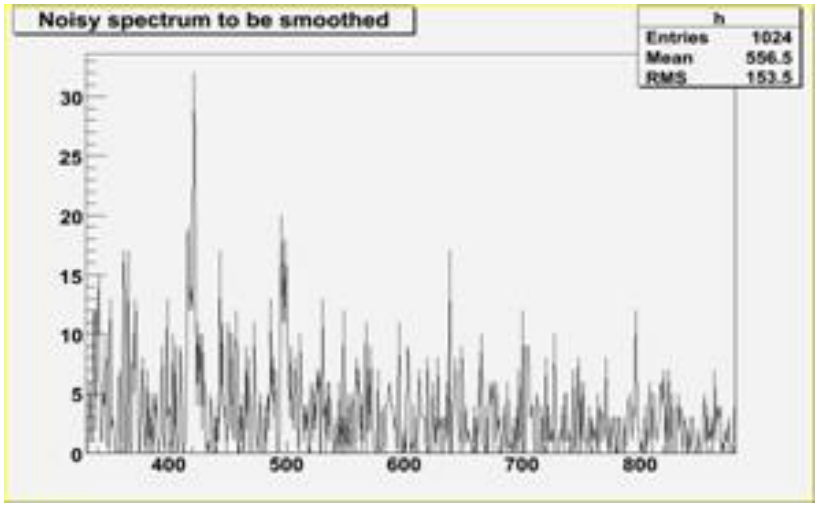
const char* SmoothMarkov(float* source, Int_t ssize, Int_t averWindow)

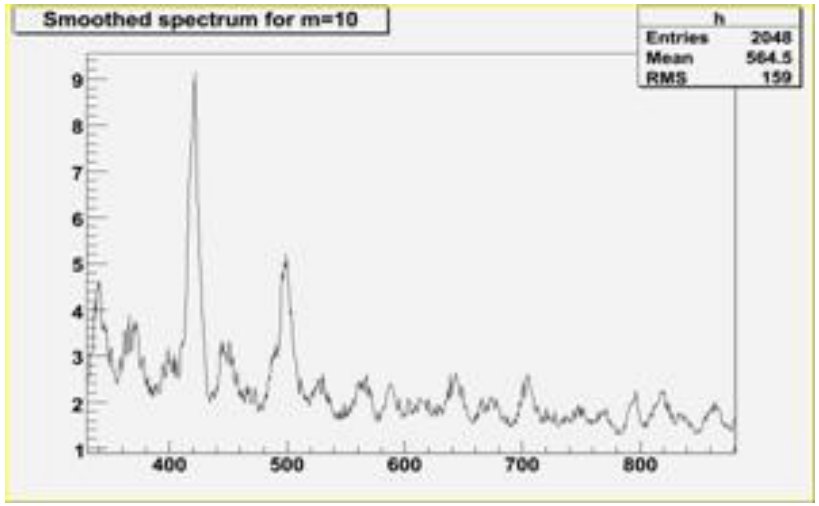
Parameters:

source: pointer to the array of source spectrum

ssize: length of source array

averWindow: width of averaging smoothing window



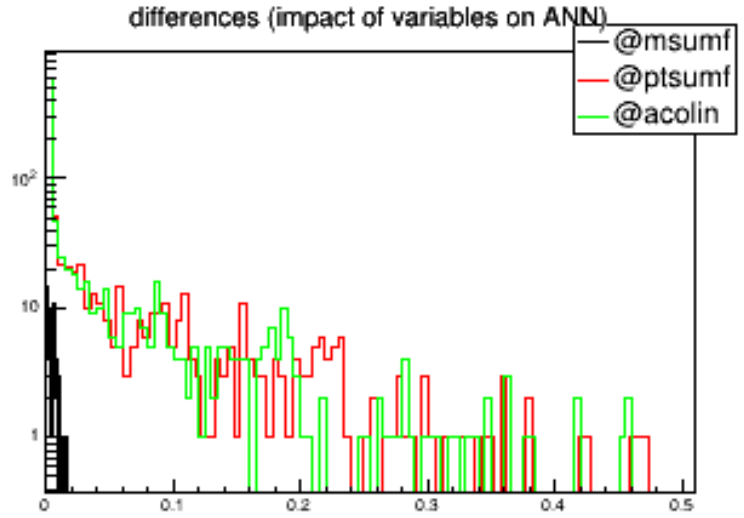


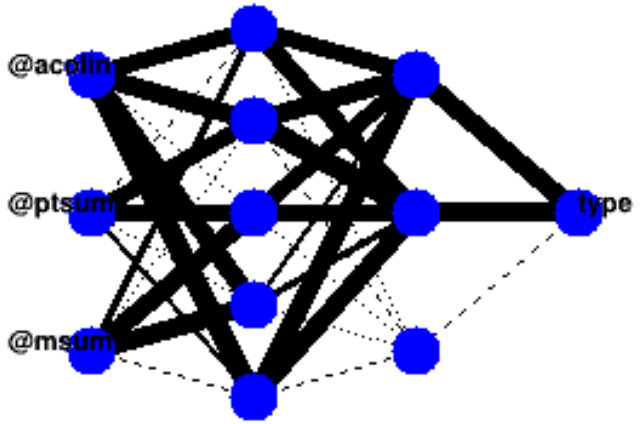
siguang@pku.edu.cn

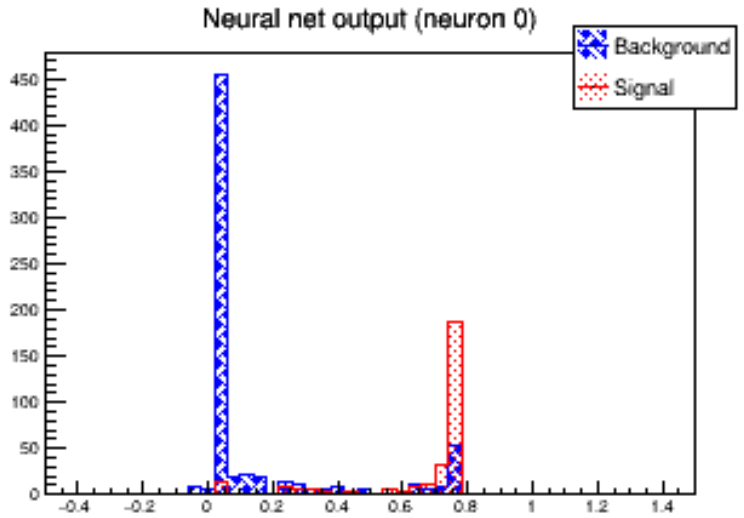
能谱光滑参考代码

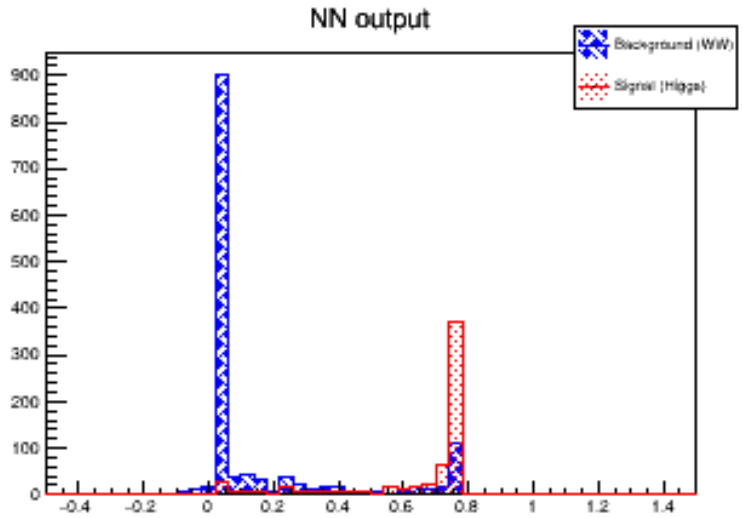
```
// Example to illustrate smoothing using Markov algorithm (class TSpectrum).
// To execute this example, do
// root > .x Smoothing.C
void Smoothing() {
   Int_t i;
   Double_t nbins = 1024;
   Double_t xmin  = 0;
   Double_t xmax  = (Double_t)nbins;
   Float_t * source = new float[nbins];
   TH1F *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");
}
```

tutorials/mlp/mlpHiggs.C









siguang@pku.edu.cn

```
void mlpHiggs(Int_t ntrain=100) {
// Example of a Multi Layer Perceptron
// For a LEP search for invisible Higgs boson, a neural network
// was used to separate the signal from the background passing
// some selection cuts. Here is a simplified version of this network,
// taking into account only WW events.
//Author: Christophe Delaere

   if (!gROOT->GetClass("TMultiLayerPerceptron")) {
      gSystem->Load("libMLP");
   }

   // Prepare inputs
   // The 2 trees are merged into one, and a "type" branch,
   // equal to 1 for the signal and 0 for the background is added.
   const char *fname = "mlpHiggs.root";
   TFile *input = 0;
   if (!gSystem->AccessPathName(fname)) {
      input = TFile::Open(fname);
   } else {
      printf("accessing %s file from http://root.cern.ch/files\n",fname);
      input = TFile::Open(Form("http://root.cern.ch/files/%s",fname));
   }
```

```
if (!input) return;

TTree *signal = (TTree *) input->Get("sig_filtered");
TTree *background = (TTree *) input->Get("bg_filtered");
TTree *simu = new TTree("MonteCarlo", "Filtered Monte Carlo Events");
Float_t ptsumf, qelep, nch, msumf, minvis, acopl, acolin;
Int_t type;
signal->SetBranchAddress("ptsumf", &ptsumf);
signal->SetBranchAddress("qelep",  &qelep);
signal->SetBranchAddress("nch",    &nch);
signal->SetBranchAddress("msumf",  &msumf);
signal->SetBranchAddress("minvis", &minvis);
signal->SetBranchAddress("acopl",  &acopl);
signal->SetBranchAddress("acolin", &acolin);
background->SetBranchAddress("ptsumf", &ptsumf);
background->SetBranchAddress("qelep",  &qelep);
background->SetBranchAddress("nch",    &nch);
background->SetBranchAddress("msumf",  &msumf);
background->SetBranchAddress("minvis", &minvis);
background->SetBranchAddress("acopl",  &acopl);
```

```
background->SetBranchAddress("acolin", &acolin);
simu->Branch("ptsumf", &ptsumf, "ptsumf/F");
simu->Branch("qelep",  &qelep,  "qelep/F");
simu->Branch("nch",    &nch,    "nch/F");
simu->Branch("msumf",  &msumf,  "msumf/F");
simu->Branch("minvis", &minvis, "minvis/F");
simu->Branch("acopl",  &acopl,  "acopl/F");
simu->Branch("acolin", &acolin, "acolin/F");
simu->Branch("type",   &type,   "type/I");
type = 1;
Int_t i;
for (i = 0; i < signal->GetEntries(); i++) {
   signal->GetEntry(i);
   simu->Fill();
}
type = 0;
for (i = 0; i < background->GetEntries(); i++) {
   background->GetEntry(i);
   simu->Fill();
}
```

```
// Build and train the NN ptsumf is used as a weight since we are primarly
// interested  by high pt events.
// The datasets used here are the same as the default ones.
TMultiLayerPerceptron *mlp =
   new TMultiLayerPerceptron("@msumf,@ptsumf,@acolin:5:3:type",
                             "ptsumf",simu,"Entry$%2","(Entry$+1)%2");
mlp->Train(ntrain, "text,graph,update=10");
//   mlp->Export("test","python");
mlp->Export("test","C++");
// Use TMLPAnalyzer to see what it looks for
TCanvas* mlpa_canvas = new TCanvas("mlpa_canvas","Network analysis");
mlpa_canvas->Divide(2,2);
TMLPAnalyzer ana(mlp);
// Initialisation
ana.GatherInformations();
// output to the console
ana.CheckNetwork();
mlpa_canvas->cd(1);
// shows how each variable influences the network
ana.DrawDInputs();
```

Added for C++ output

```
.............
// Use the NN to plot the results for each sample
// This will give approx. the same result as DrawNetwork.
// All entries are used, while DrawNetwork focuses on
// the test sample. Also the xaxis range is manually set.
TH1F *bg = new TH1F("bgh", "NN output", 50, -.5, 1.5);
TH1F *sig = new TH1F("sigh", "NN output", 50, -.5, 1.5);
bg->SetDirectory(0);
sig->SetDirectory(0);
Double_t params[3];
for (i = 0; i < background->GetEntries(); i++) {
   background->GetEntry(i);
   params[0] = msumf;
   params[1] = ptsumf;
   params[2] = acolin;
   bg->Fill(mlp->Evaluate(0, params));
}
for (i = 0; i < signal->GetEntries(); i++) {
   signal->GetEntry(i);
   params[0] = msumf;
   params[1] = ptsumf;
   params[2] = acolin;
   sig->Fill(mlp->Evaluate(0,params));
}
```

How to use the trained mlp

test.h and test.cxx will be created

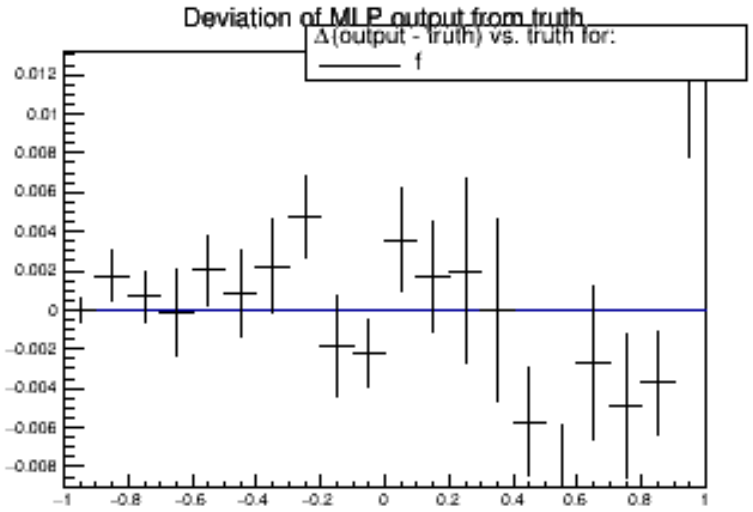
test.h

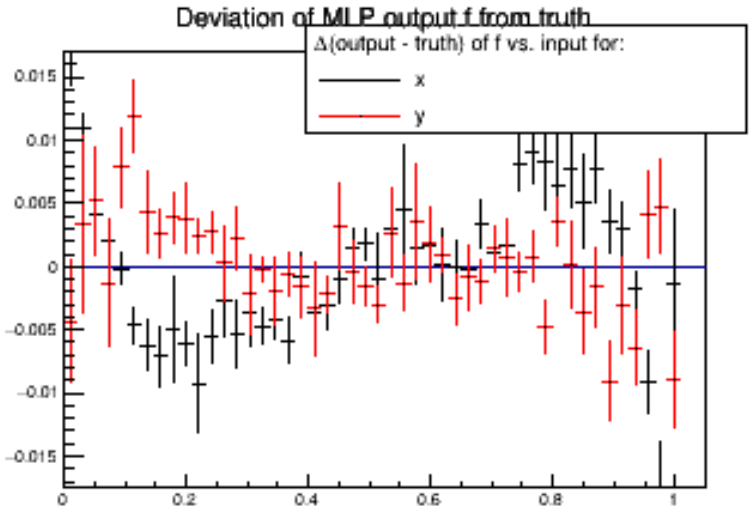
```
#ifndef test_h
#define test_h

class test {
public:
   test() {}
   ~test() {}
   double Value(int index,double in0,double in1,double
   double Value(int index, double* input);
private:
   double input0;
   double input1;
   double input2;
   double neuron0x22280c0();
   double neuron0x2228400();
   double neuron0x2239390();
   double input0x2239770();
   double neuron0x2239770();
   double input0x2239aa0();
   ..............
   double synapse0x223b7f0();
   double synapse0x223b830();
   double synapse0x223b870();
   double synapse0x223bad0();
   double synapse0x223bb10();
   double synapse0x223bb50();
};

#endif // test_h
```

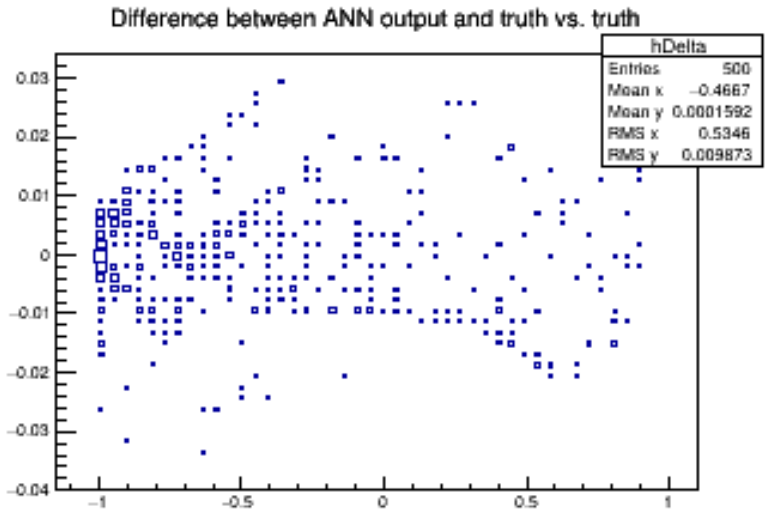
test.cxx

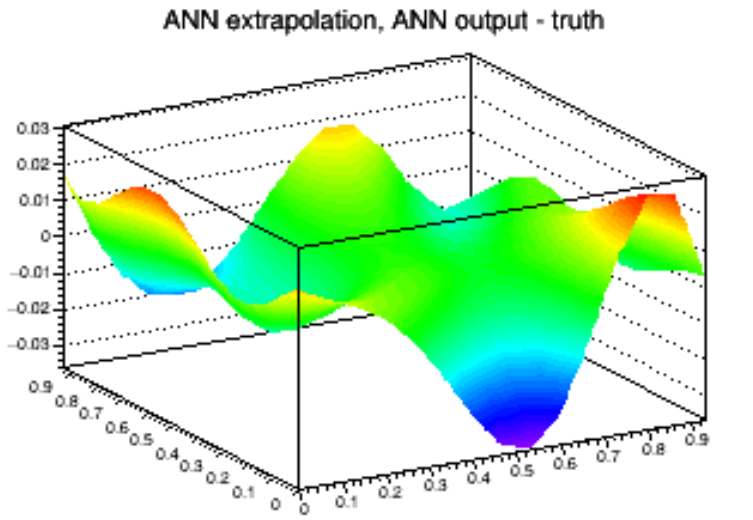
```
#include "test.h"
#include <cmath>

double test::Value(int index,double in0,double in1,double in2) {
   input0 = (in0 - 0.459987)/0.0509152;
   input1 = (in1 - 0.188581)/0.0656804;
   input2 = (in2 - 134.719)/16.5033;
   switch(index) {
     case 0:
         return neuron0x223b8b0();
     default:
         return 0.;
   }
}

double test::input0x223b8b0() {
   double input = 1.45517;
   input += synapse0x223bad0();
   input += synapse0x223bb10();
   input += synapse0x223bb50();
   return input;
}

double test::synapse0x223bad0() {
   return (neuron0x223aaa0()*2.58206);
}
```

/tutorials/mlp/mlpRegression.C









siguang@pku.edu.cn

```
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;
   for (Int_t i=0; i<1000; i++) {
      Float_t x=r.Rndm();
      Float_t y=r.Rndm();
      // 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:10:8:f",t,
   "Entry$%2","(Entry$%2)==0");
mlp->Train(150,"graph update=10");

// analyze it
TMLPAnalyzer* mlpa=new TMLPAnalyzer(mlp);
mlpa->GatherInformations();
mlpa->CheckNetwork();
mlpa->DrawDInputs();

// draw statistics shows the quality of the ANN's approximation
TCanvas* cIO=new TCanvas("TruthDeviation", "TruthDeviation");
cIO->Divide(2,2);
cIO->cd(1);
// draw the difference between the ANN's output for (x,y) and
// the true value f(x,y), vs. f(x,y), as TProfiles
mlpa->DrawTruthDeviations();
```

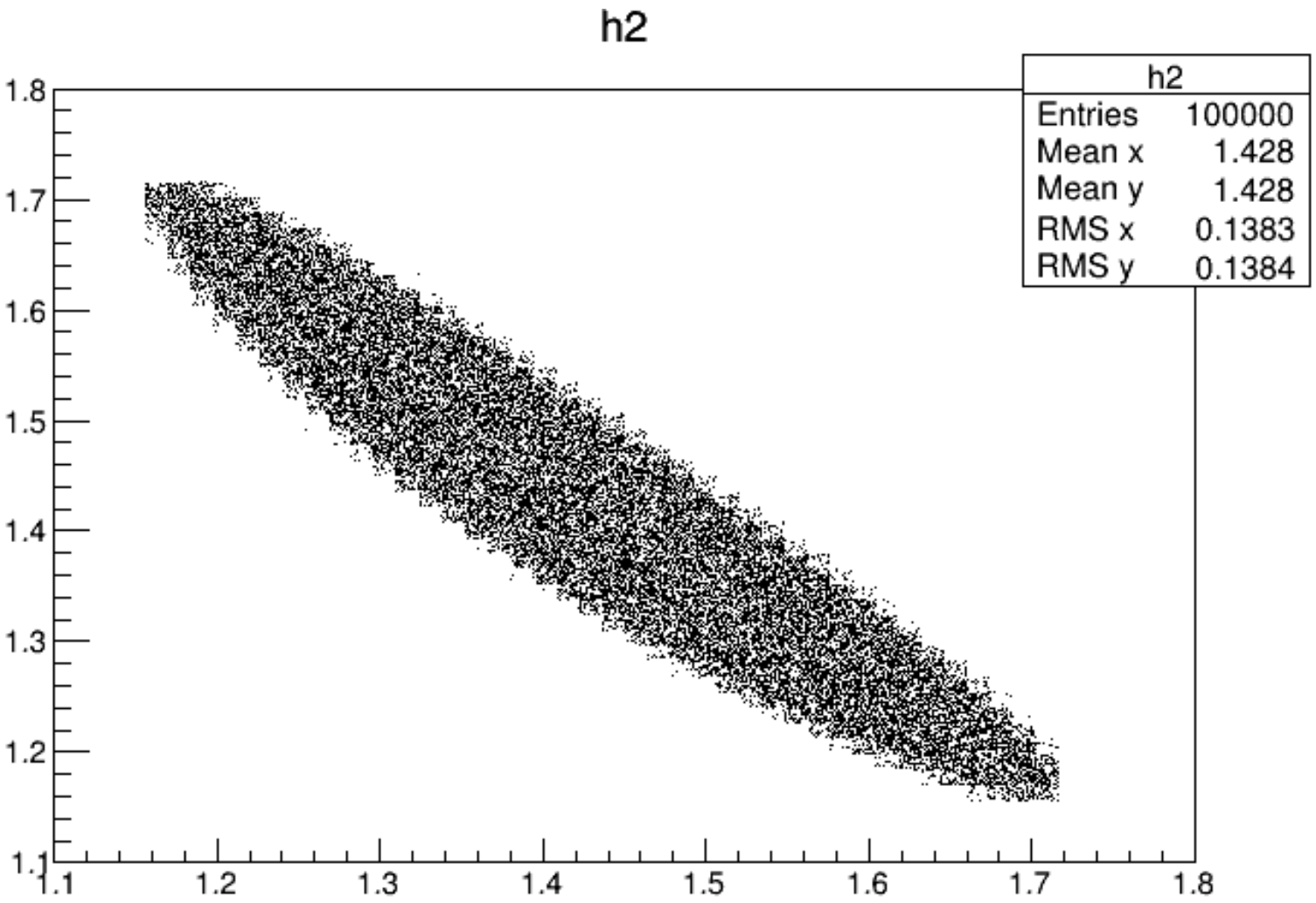
```
cIO->cd(2);
// draw the difference between the ANN's output for (x,y) and
// the true value f(x,y), vs. x, and vs. y, as TProfiles
mlpa->DrawTruthDeviationInsOut();

cIO->cd(3);
// draw a box plot of the ANN's output for (x,y) vs f(x,y)
mlpa->GetIOTree()->Draw("Out.Out0-True.True0:True.True0>>hDelta","","goff");
TH2F* hDelta=(TH2F*)gDirectory->Get("hDelta");
hDelta->SetTitle("Difference between ANN output and truth vs. truth");
hDelta->Draw("BOX");

cIO->cd(4);
// draw difference of ANN's output for (x,y) vs f(x,y) assuming
// the ANN can extrapolate
Double_t vx[225];
Double_t vy[225];
Double_t delta[225];
```

```
Double_t v[2];
for (Int_t ix=0; ix<15; ix++) {
  //     v[0]=ix/5.-1.;//out of the trainning range, commented out by siguang
  v[0]=ix/15.;
   for (Int_t iy=0; iy<15; iy++) {
     //   v[1]=iy/5.-1.;  //out of the trainning range
     v[1]=iy/15.;
      Int_t idx=ix*15+iy;
      vx[idx]=v[0];
      vy[idx]=v[1];
      delta[idx]=mlp->Evaluate(0, v)-theUnknownFunction(v[0],v[1]);
   }
}
```

```
   TGraph2D* g2Extrapolate=new TGraph2D("ANN extrapolation",
                                        "ANN extrapolation, ANN output - truth",
                                        225, vx, vy, delta);
   g2Extrapolate->Draw("TRI2");
}
```

PhaseSpace

$ROOTSYS/tutorials/physics/PhaseSpace.C



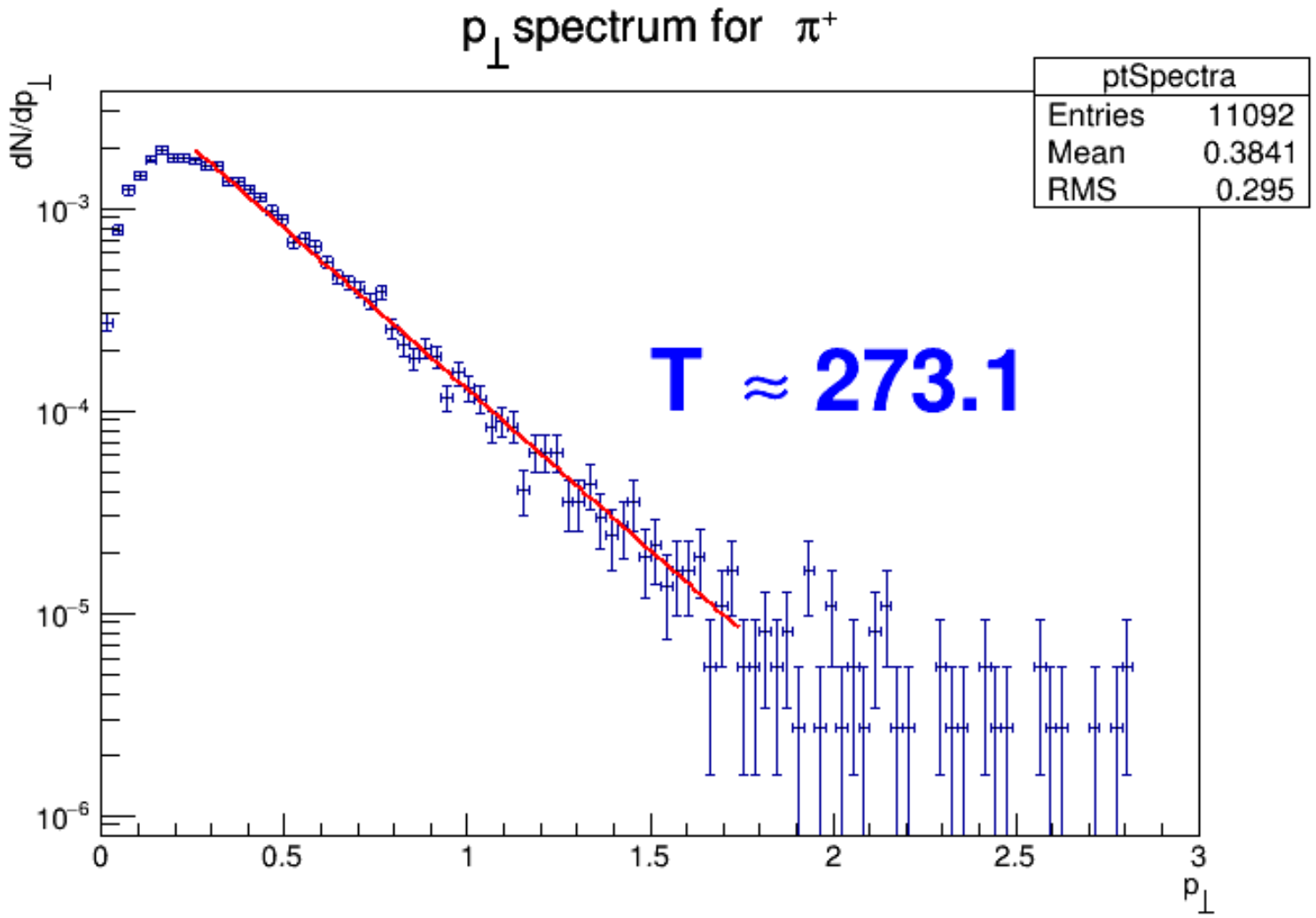
siguang@pku.edu.cn

$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();
}
```

$\gamma + H \rightarrow p + \pi^+ + \pi^-$

/tutorials/pythia$ root pythiaExample.C



siguang@pku.edu.cn

```
//
// Using Pythia6 with ROOT
// To make an event sample (of size 100) do
//
//    shell> root
//    root [0] .L pythiaExample.C
//    root [1] makeEventSample(1000)
//
// To start the tree view on the generated tree, do
//
//    shell> root
//    root [0] .L pythiaExample.C
//    root [1] showEventSample()
//
//
// The following session:
//    shell> root
//    root [0] .x pythiaExample.C(500)
// will execute makeEventSample(500) and showEventSample()
//
// Alternatively, you can compile this to a program
// and then generate 1000 events with
//
//    ./pythiaExample 1000
//
```

```
// For the licensing terms see $ROOTSYS/LICENSE.
//
#ifndef __CINT__
#include "TApplication.h"
#include "TPythia6.h"
#include "TFile.h"
#include "TError.h"
#include "TTree.h"
#include "TClonesArray.h"
#include "TH1.h"
#include "TF1.h"
#include "TStyle.h"
#include "TLatex.h"
#include "TCanvas.h"
#include "Riostream.h"
#include <cstdlib>
using namespace std;
#endif

#define FILENAME   "pythia.root"
#define TREENAME   "tree"
#define BRANCHNAME "particles"
#define HISTNAME   "ptSpectra"
#define PDGNUMBER  211
```

```
// This function just load the needed libraries if we're executing from
// an interactive session.
void loadLibraries()
{
#ifdef __CINT__
  // Load the Event Generator abstraction library, Pythia 6
  // library, and the Pythia 6 interface library.
  gSystem->Load("libEG");
  gSystem->Load("$ROOTSYS/../pythia6/libPythia6"); //change to your setup
  gSystem->Load("libEGPythia6");
#endif
}

// nEvents is how many events we want.
int makeEventSample(Int_t nEvents)
{
  // Load needed libraries
  loadLibraries();

  // Create an instance of the Pythia event generator ...
  TPythia6* pythia = new TPythia6;

  // ... and initialise it to run p+p at sqrt(200) GeV in CMS
  pythia->Initialize("cms", "p", "p", 200);

  // Open an output file
  TFile* file = TFile::Open(FILENAME, "RECREATE");
  if (!file || !file->IsOpen()) {
    Error("makeEventSample", "Couldn;t open file %s", FILENAME);
    return 1;
  }
```

指向 libPythia6.so所在的库或已经设置了正确的环境
export PYTHIA6=/home/wsg/work/pythia6/pythia6428
export LD_LIBRARY_PATH=$ROOTSYS/lib/root:$PYTHIA6:$PYTHIA8/lib:$LD_LIBRARY_PATH

```
// Make a tree in that file ...
TTree* tree = new TTree(TREENAME, "Pythia 6 tree");

// ... and register a the cache of pythia on a branch (It's a
// TClonesArray of TMCParticle objects. )
TClonesArray* particles = (TClonesArray*)pythia->GetListOfParticles();
tree->Branch(BRANCHNAME, &particles);

// Now we make some events
for (Int_t i = 0; i < nEvents; i++) {
  // Show how far we got every 100'th event.
  if (i % 100 == 0)
    cout << "Event # " << i << endl;

  // Make one event.
  pythia->GenerateEvent();

  // Maybe you want to have another branch with global event
  // information.  In that case, you should process that here.
  // You can also filter out particles here if you want.

  // Now we're ready to fill the tree, and the event is over.
  tree->Fill();
}

// Show tree structure
tree->Print();
```

```
  // Show tree structure
  tree->Print();

  // After the run is over, we may want to do some summary plots:
  TH1D* hist = new TH1D(HISTNAME, "p_{#perp}  spectrum for  #pi^{+}",
                        100, 0, 3);
  hist->SetXTitle("p_{#perp}");
  hist->SetYTitle("dN/dp_{#perp}");
  char expression[64];
  sprintf(expression,"sqrt(pow(%s.fPx,2)+pow(%s.fPy,2))>>%s",
          BRANCHNAME, BRANCHNAME, HISTNAME);
  char selection[64];
  sprintf(selection,"%s.fKF==%d", BRANCHNAME, PDGNUMBER);
  tree->Draw(expression,selection);

  // Normalise to the number of events, and the bin sizes.
  hist->Sumw2();
  hist->Scale(3 / 100. / hist->Integral());
  hist->Fit("expo", "Q0+", "", .25, 1.75);
  TF1* func = hist->GetFunction("expo");
  func->SetParNames("A", "- 1 / T");
  // and now we flush and close the file
  file->Write();
  file->Close();

  return 0;
}
```

```
// Show the Pt spectra, and start the tree viewer.
int showEventSample()
{
  // Load needed libraries
  loadLibraries();

  // Open the file
  TFile* file = TFile::Open(FILENAME, "READ");
  if (!file || !file->IsOpen()) {
    Error("showEventSample", "Couldn;t open file %s", FILENAME);
    return 1;
  }

  // Get the tree
  TTree* tree = (TTree*)file->Get(TREENAME);
  if (!tree) {
    Error("showEventSample", "couldn't get TTree %s", TREENAME);
    return 2;
  }

  // Start the viewer.
  tree->StartViewer();

  // Get the histogram
  TH1D* hist = (TH1D*)file->Get(HISTNAME);
  if (!hist) {
    Error("showEventSample", "couldn't get TH1D %s", HISTNAME);
    return 4;
  }
```

```
  // Draw the histogram in a canvas
  gStyle->SetOptStat(1);
  TCanvas* canvas = new TCanvas("canvas", "canvas");
  canvas->SetLogy();
  hist->Draw("e1");
  TF1* func = hist->GetFunction("expo");

  char expression[64];
  sprintf(expression,"T #approx %5.1f", -1000 / func->GetParameter(1));
  TLatex* latex = new TLatex(1.5, 1e-4, expression);
  latex->SetTextSize(.1);
  latex->SetTextColor(4);
  latex->Draw();

  return 0;
}

void pythiaExample(Int_t n=1000) {
   makeEventSample(n);
   showEventSample();
}
```

```
#ifndef __CINT__
int main(int argc, char** argv)
{
  TApplication app("app", &argc, argv);

  Int_t n = 100;
  if (argc > 1)
    n = strtol(argv[1], NULL, 0);

  int retVal = 0;
  if (n > 0)
    retVal = makeEventSample(n);
  else {
    retVal = showEventSample();
    app.Run();
  }

  return retVal;
}
#endif
```

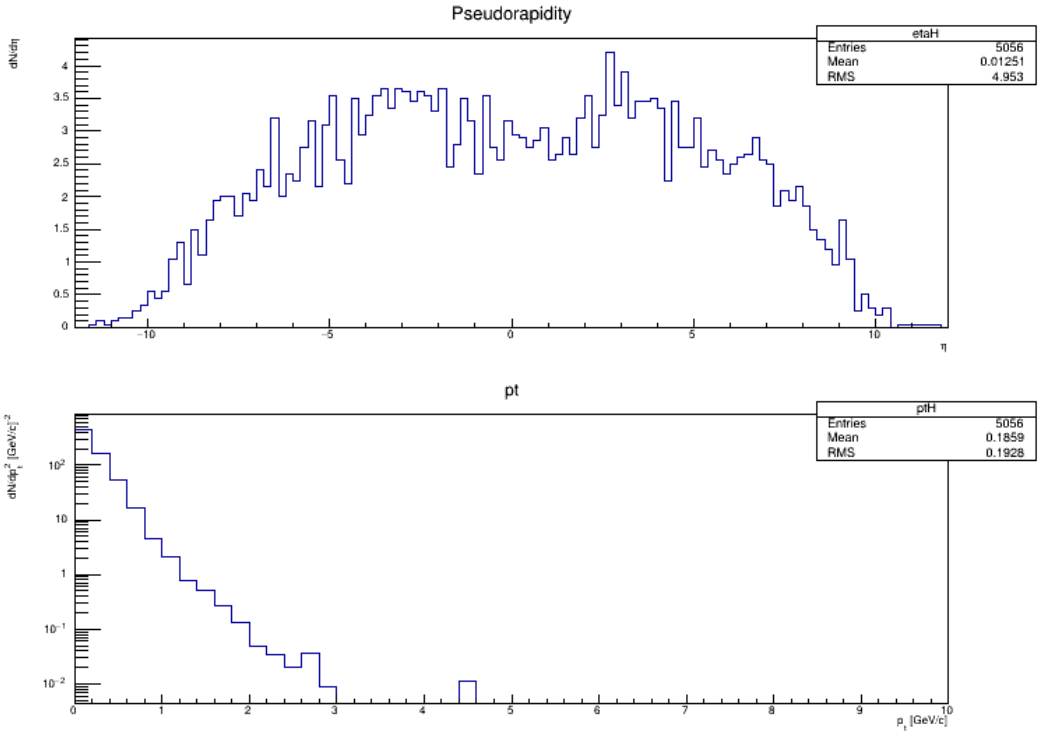
```
g++ -o pythiaExample pythiaExample.C `root-config --cflags --libs`  -lEG -lEGPythia6
  -L/home/wsg/work/pythia6/pythia6428 -lPythia6
./pythiaExample 100
```

```
*.....................................................................*
*Br   20 :particles.fLifetime : Float_t fLifetime[particles_]         *
*Entries :      100 : Total  Size=      44780 bytes  File Size  =      6464 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=   4.84     *
*.....................................................................*
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
```

/tutorials/pythia/pythia8.C

```
// Configure    following 3 lines can not work in root534
//   pythia8->ReadString("SoftQCD:minBias = on");
//   pythia8->ReadString("SoftQCD:singleDiffractive = on");
//   pythia8->ReadString("SoftQCD:doubleDiffractive = on");
// Configure    using next line!  siguang wang
   pythia8->ReadString("HardQCD:all = on");
```

原来的自带程序错误！



siguang@pku.edu.cn

```
// pythia8 basic example
//Author: Andreas Morsch
//
// to run, do
//  root > .x pythia8.C
//
// Note that before executing this script,
//   -the env variable PYTHIA8 must point to the pythia8100 (or newer) directory
//   -the env variable PYTHIA8DATA must be defined and it must point to $PYTHIA8/xmldoc
//
void pythia8(Int_t nev  = 100, Int_t ndeb = 1)
{
   char *p8dataenv = gSystem->Getenv("PYTHIA8DATA");
   if (!p8dataenv) {
      char *p8env = gSystem->Getenv("PYTHIA8");

      if (!p8env) {
         Error("pythia8.C",
               "Environment variable PYTHIA8 must contain path to pythia directory!");
         return;
      }
      TString p8d = p8env;
      p8d += "/xmldoc";
      gSystem->Setenv("PYTHIA8DATA", p8d);
   }
```

```
char* path = gSystem->ExpandPathName("$PYTHIA8DATA");
if (gSystem->AccessPathName(path)) {
      Error("pythia8.C",
            "Environment variable PYTHIA8DATA must contain path to $PYTHIA8/xmldoc directory !");
   return;
}
```

```
// Load libraries
#ifndef G__WIN32 // Pythia8 is a static library on Windows
   gSystem->Load("$PYTHIA8/lib/libpythia8");
#endif
   gSystem->Load("libEG");
   gSystem->Load("libEGPythia8");
// Histograms
   TH1F* etaH = new TH1F("etaH", "Pseudorapidity", 120, -12., 12.);
   TH1F* ptH  = new TH1F("ptH",  "pt",              50,   0., 10.);
```

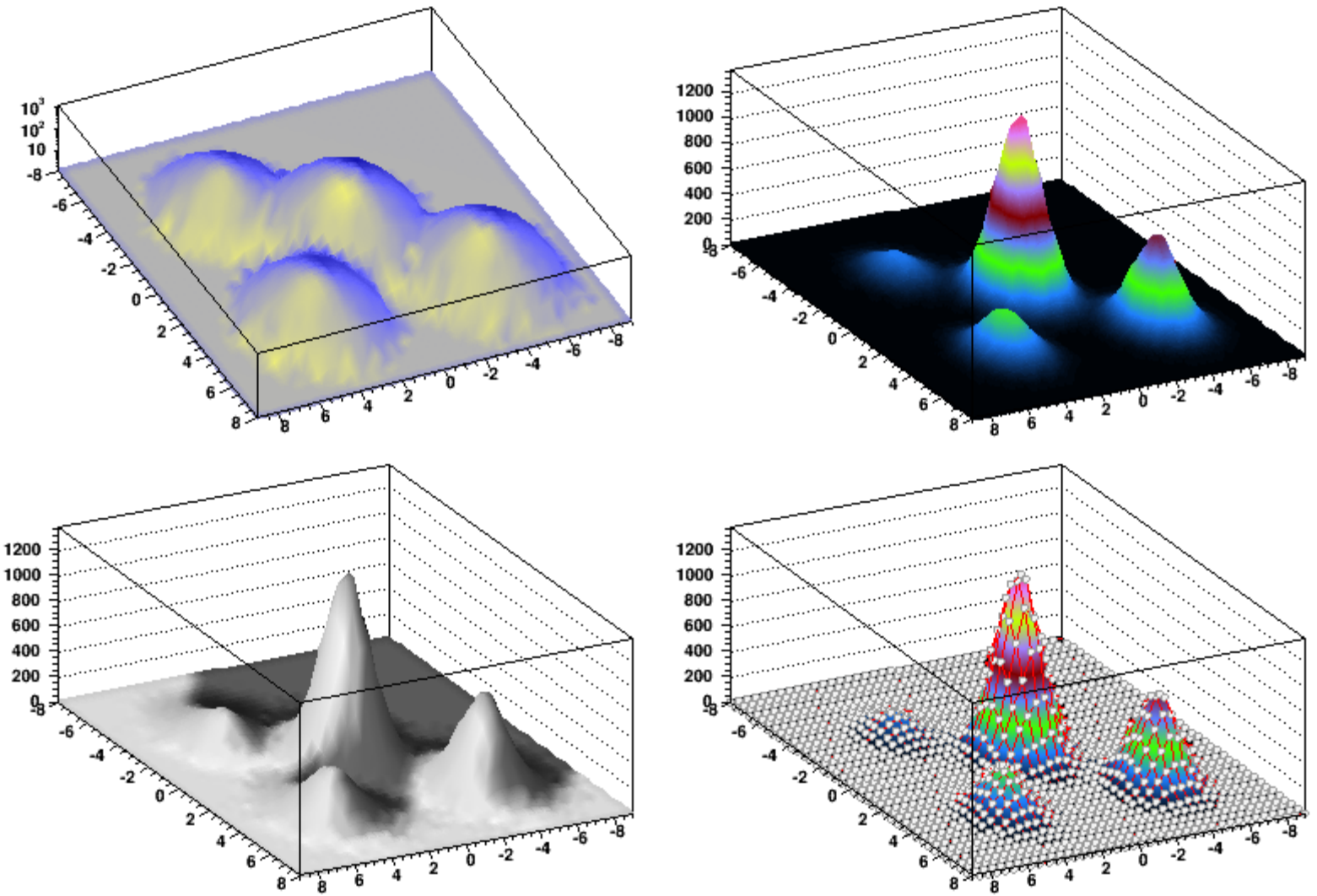
```
// Array of particles
   TClonesArray* particles = new TClonesArray("TParticle", 1000);
// Create pythia8 object
   TPythia8* pythia8 = new TPythia8();

// Configure    following 3 lines can not work in root534
//   pythia8->ReadString("SoftQCD:minBias = on");
//   pythia8->ReadString("SoftQCD:singleDiffractive = on");
//   pythia8->ReadString("SoftQCD:doubleDiffractive = on");
// Configure    using next line!  siguang wang
   pythia8->ReadString("HardQCD:all = on");
```

```
// Initialize

   pythia8->Initialize(2212 /* p */, 2212 /* p */, 14000. /* TeV */);

// Event loop
   for (Int_t iev = 0; iev < nev; iev++) {
      pythia8->GenerateEvent();
      if (iev < ndeb) pythia8->EventListing();
      pythia8->ImportParticles(particles,"All");
      Int_t np = particles->GetEntriesFast();
// Particle loop
      for (Int_t ip = 0; ip < np; ip++) {
         TParticle* part = (TParticle*) particles->At(ip);
         Int_t ist = part->GetStatusCode();
         // Positive codes are final particles.
         if (ist <= 0) continue;
         Int t pdg = part->GetPdgCode();

         Float_t charge = TDatabasePDG::Instance()->GetParticle(pdg)->Charge();
         if (charge == 0.) continue;
         Float_t eta = part->Eta();
         Float_t pt  = part->Pt();

         etaH->Fill(eta);
         if (pt > 0.) ptH->Fill(pt, 1./(2. * pt));
      }
   }
```

```
  pythia8->PrintStatistics();

  TCanvas* c1 = new TCanvas("c1","Pythia8 test example",800,800);
  c1->Divide(1, 2);
  c1->cd(1);
  etaH->Scale(5./Float_t(nev));
  etaH->Draw();
  etaH->SetXTitle("#eta");
  etaH->SetYTitle("dN/d#eta");

  c1->cd(2);
  gPad->SetLogy();
  ptH->Scale(5./Float_t(nev));
  ptH->Draw();
  ptH->SetXTitle("p_{t} [GeV/c]");
  ptH->SetYTitle("dN/dp_{t}^{2} [GeV/c]^{-2}");
}
```

```
*-------  PYTHIA Event and Cross Section Statistics  -------------------------------------------------------*
|                                                                                                           |
| Subprocess                                    Code |            Number of events        |      sigma +- delta    |
|                                                    |       Tried   Selected   Accepted  |     (estimated) (mb)   |
|                                                    |                                    |                        |
|---------------------------------------------------------------------------------------------------------|
|                                                    |                                    |                        |
| g g -> g g                                     111 |        6329        670        670  |   2.143e+03  4.212e+01 |
| g g -> q qbar (uds)                            112 |          39         10         10  |   2.248e+01  3.929e+00 |
| q g -> q g                                     113 |        2417        288        288  |   9.224e+02  3.077e+01 |
| q q(bar)' -> q q(bar)'                         114 |         358         30         30  |   1.029e+02  8.143e+00 |
| q qbar -> g g                                  115 |           1          0          0  |   0.000e+00  0.000e+00 |
| q qbar -> q' qbar' (uds)                       116 |           0          0          0  |   0.000e+00  0.000e+00 |
| g g -> c cbar                                  121 |          30          2          2  |   4.457e+00  1.693e+00 |
| q qbar -> c cbar                               122 |           0          0          0  |   0.000e+00  0.000e+00 |
| g g -> b bbar                                  123 |           0          0          0  |   0.000e+00  0.000e+00 |
| q qbar -> b bbar                               124 |           0          0          0  |   0.000e+00  0.000e+00 |
|                                                    |                                    |                        |
| sum                                                |        9174       1000       1000  |   3.195e+03  5.297e+01 |
|                                                                                                           |
*-------  End PYTHIA Event and Cross Section Statistics -------------------------------------------------------*

*-------  PYTHIA Error and Warning Messages Statistics  ----------------------------------------------------*
|                                                                                                           |
|  times   message                                                                                          |
|                                                                                                           |
|      1   Warning in PhaseSpace2to2tauyz::trialKin: maximum for cross section violated                     |
|                                                                                                           |
*-------  End PYTHIA Error and Warning Messages Statistics  ----------------------------------------------------*
root [2]
```

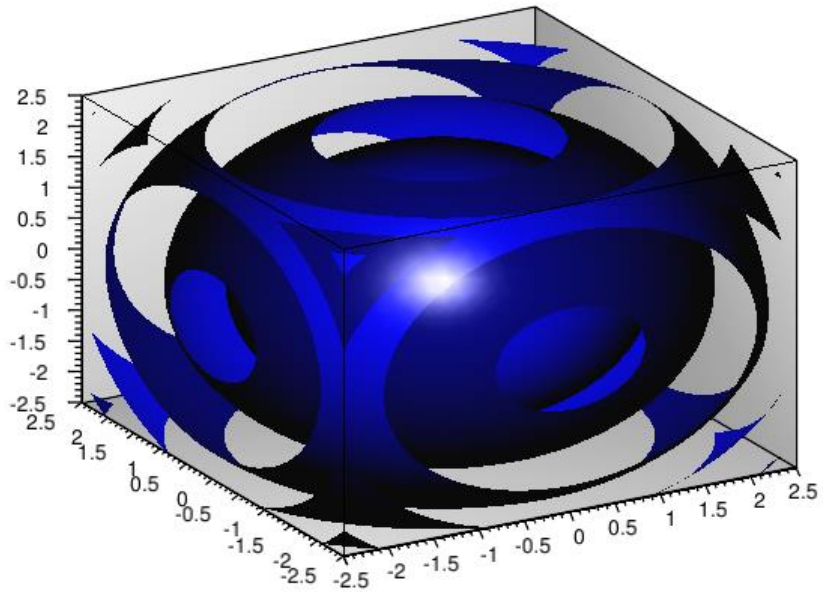
2D 显示



$ROOTSYS/tutorials/spectrum$ root spectrumpainter.C

siguang@pku.edu.cn

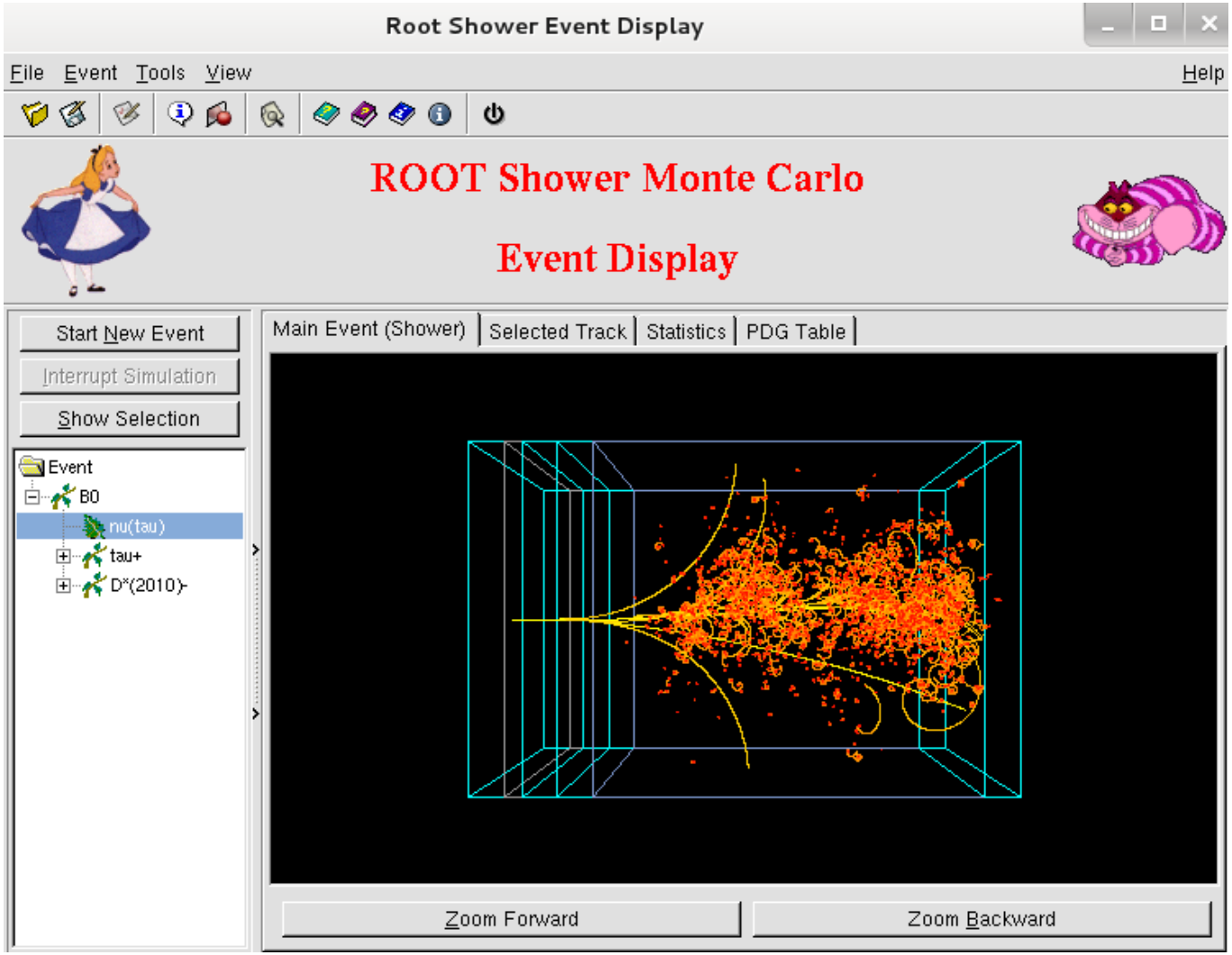
画OpenGL图



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");
}
```

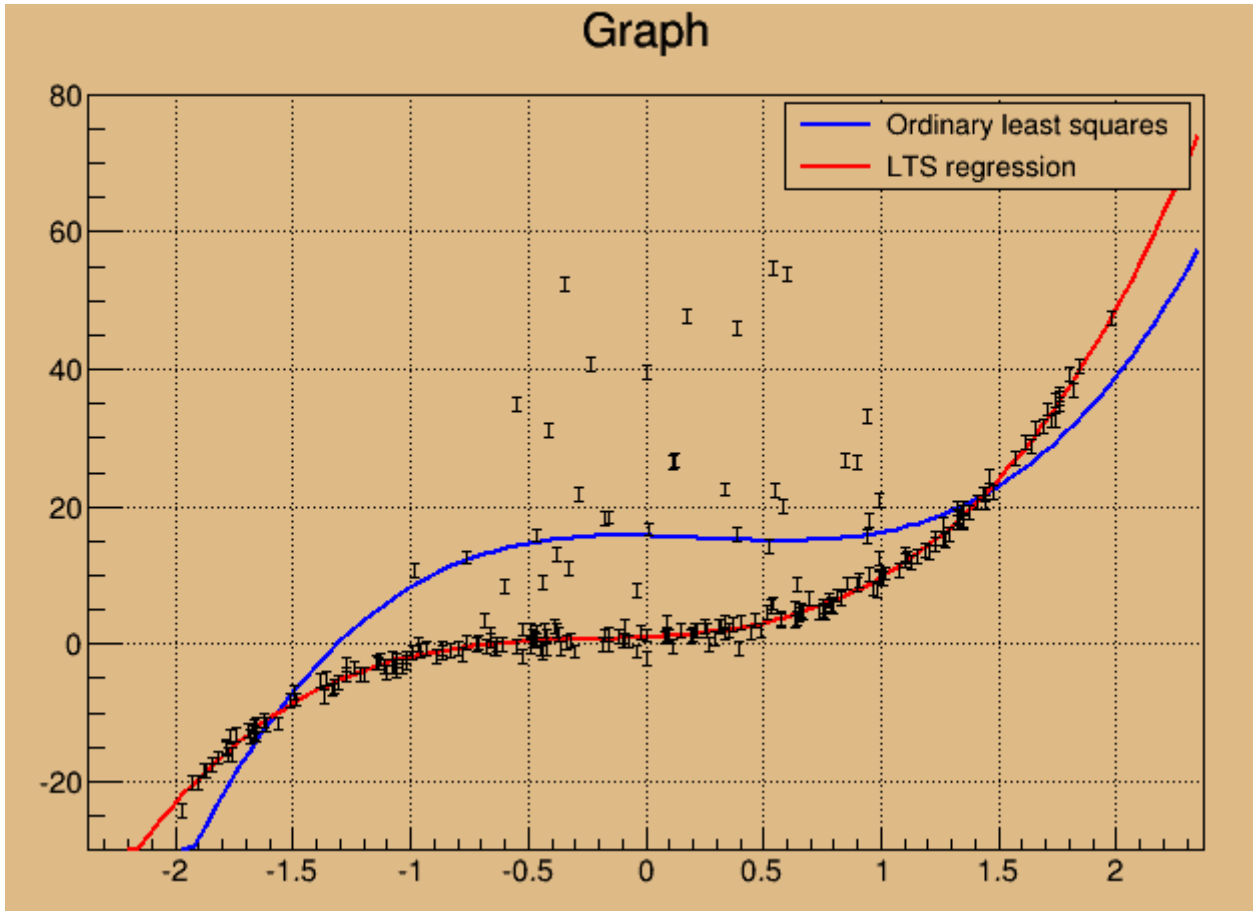
图形界面



$ROOTSYS/test/RootShower$ ./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);
  }
```

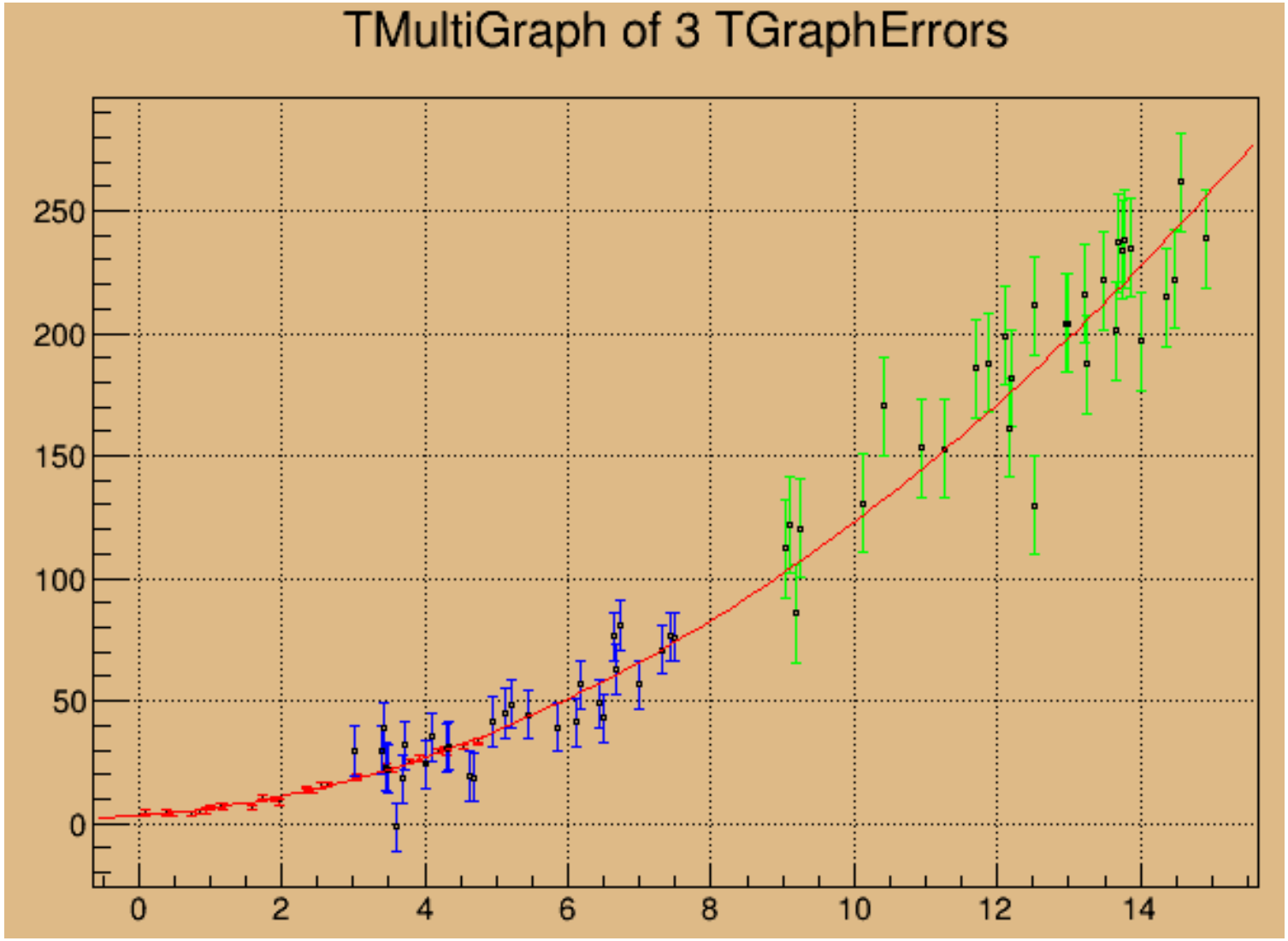
$ROOTSYS/tutorials/fit/fitLinearRobust.C

```
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



$ROOTSYS/tutorials/fit/fitMultiGraph.C

```
void fitMultiGraph()
{
   //fitting a parabola to a multigraph of 3 partly overlapping graphs
   //with different errors
   //Author: Anna Kreshuk

   Int_t n = 30;
   Double_t *x1 = new Double_t[n];
   Double_t *x2 = new Double_t[n];
   Double_t *x3 = new Double_t[n];
   Double_t *y1 = new Double_t[n];
   Double_t *y2 = new Double_t[n];
   Double_t *y3 = new Double_t[n];
   Double_t *e1 = new Double_t[n];
   Double_t *e2 = new Double_t[n];
   Double_t *e3 = new Double_t[n];
```

$ROOTSYS/tutorials/fit/fitMultiGraph.C

```
//generate the data for the graphs
TRandom r;
Int_t i;
for (i=0; i<n; i++) {
   x1[i] = r.Uniform(0.1, 5);
   x2[i] = r.Uniform(3, 8);
   x3[i] = r.Uniform(9, 15);
   y1[i] = 3 + 2*x1[i] + x1[i]*x1[i] + r.Gaus();
   y2[i] = 3 + 2*x2[i] + x2[i]*x2[i] + r.Gaus()*10;
   e1[i] = 1;
   e2[i] = 10;
   e3[i] = 20;
   y3[i] = 3 + 2*x3[i] + x3[i]*x3[i] + r.Gaus()*20;
}

//create the graphs and set their drawing options
TGraphErrors *gr1 = new TGraphErrors(n, x1, y1, 0, e1);
TGraphErrors *gr2 = new TGraphErrors(n, x2, y2, 0, e2);
TGraphErrors *gr3 = new TGraphErrors(n, x3, y3, 0, e3);
gr1->SetLineColor(kRed);
gr2->SetLineColor(kBlue);
gr2->SetMarkerStyle(24);
gr2->SetMarkerSize(0.3);
gr3->SetLineColor(kGreen);
gr3->SetMarkerStyle(24);
gr3->SetMarkerSize(0.3);
```

$ROOTSYS/tutorials/fit/fitMultiGraph.C

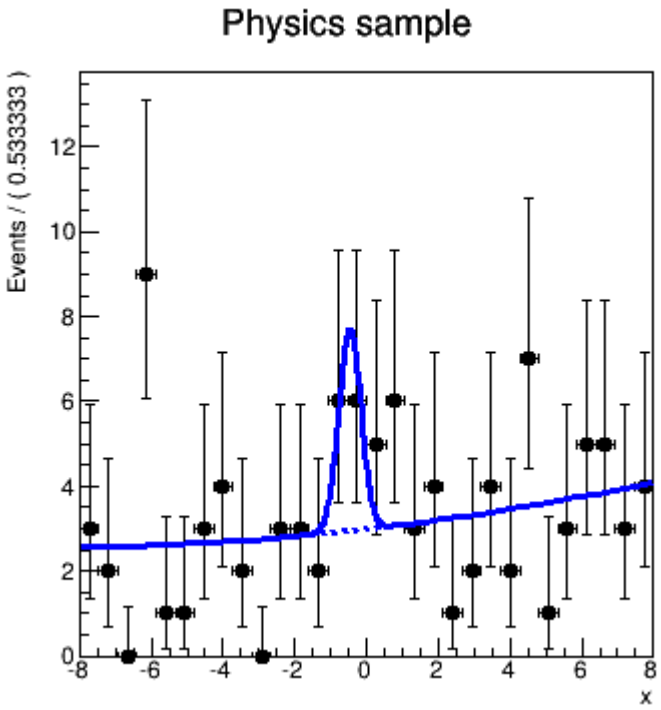
```
//add the graphs to the multigraph
TMultiGraph *mg=new TMultiGraph("mg",
   "TMultiGraph of 3 TGraphErrors");
mg->Add(gr1);
mg->Add(gr2);
mg->Add(gr3);
```

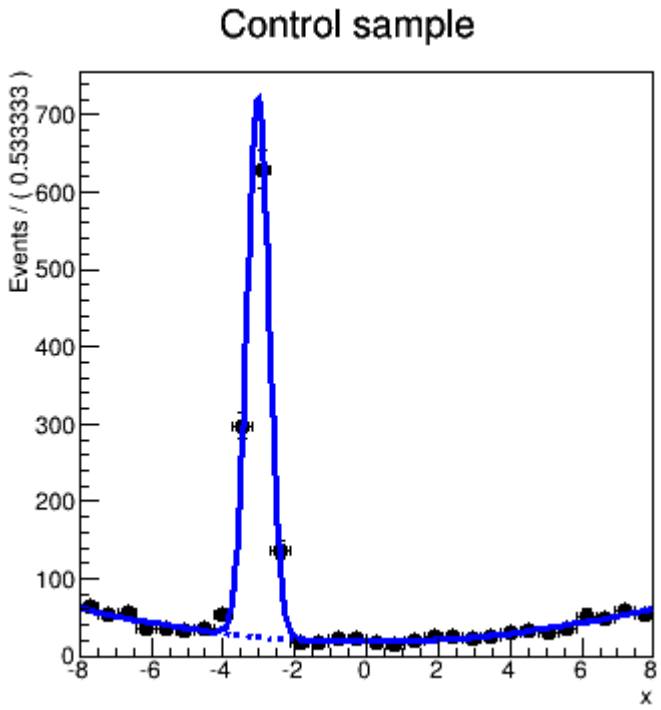
$ROOTSYS/tutorials/fit/fitMultiGraph.C

```
   TCanvas *myc = new TCanvas("myc",
      "Fitting a MultiGraph of 3 TGraphErrors");
   myc->SetFillColor(42);
   myc->SetGrid();

   mg->Draw("ap");

   //fit
   mg->Fit("pol2", "F");
   //mg->Fit("pol2");

   //access to the fit function
   TF1 *fpol = mg->GetFunction("pol2");
   fpol->SetLineWidth(1);

}
```

"F" If fitting a polN, switch to minuit fitter

$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) ;
```

共用sigma

$ROOTSYS/tutorials/roofit/rf501_simultaneouspdf.C

```
  // 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)) ;
// 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") ;



// 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) ;
```


siguang@pku.edu.cn

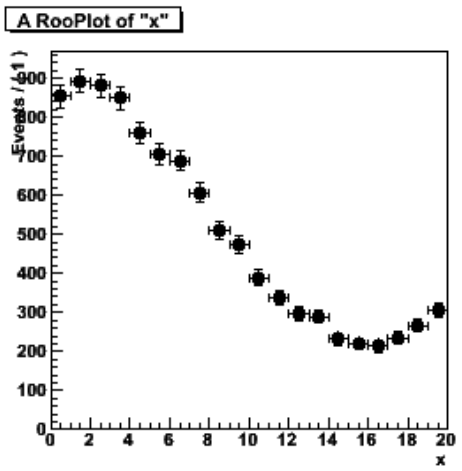
$ROOTSYS/tutorials/roofit/rf501_simultaneouspdf.C

```
  // 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(kDas
hed)) ;


    // 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() ;


}
```

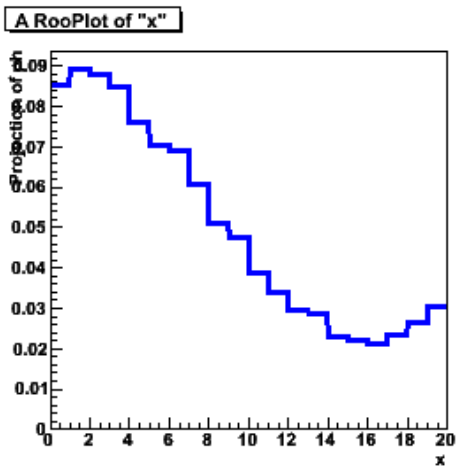
Highlight of non-parametric shapes - histograms

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

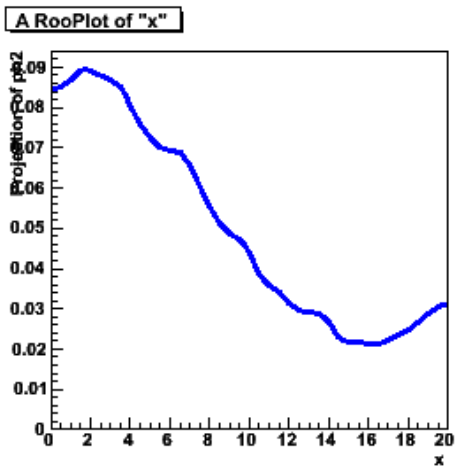
dataHist



RooHistPdf(N=0)



RooHistPdf(N=4)

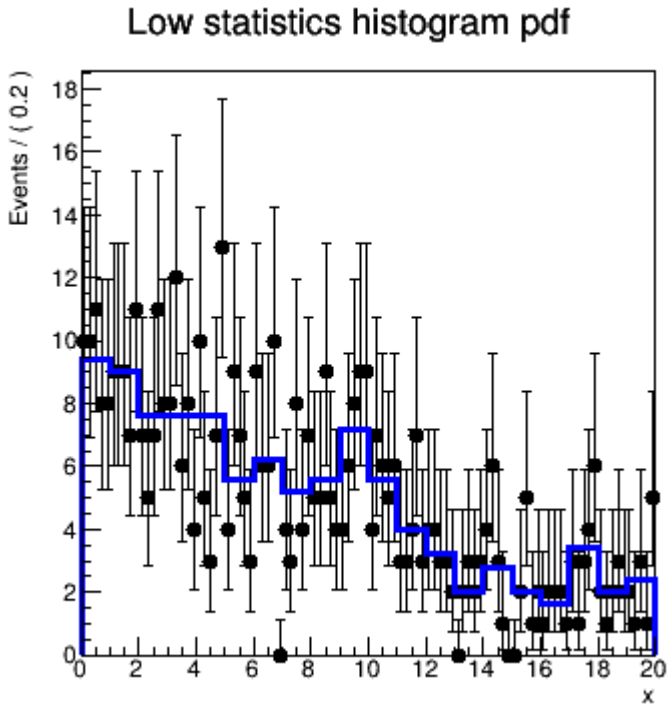


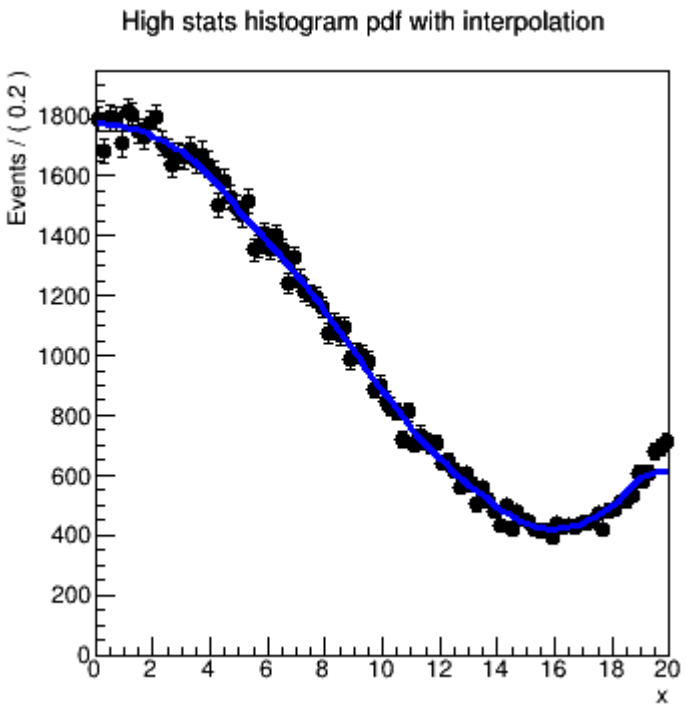

```
// Histogram based p.d.f with N-th order interpolation
RooHistPdf ph("ph","ph",x,*dataHist,N) ;
```

- 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() ;


}
```

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

Contents lists available at ScienceDirect

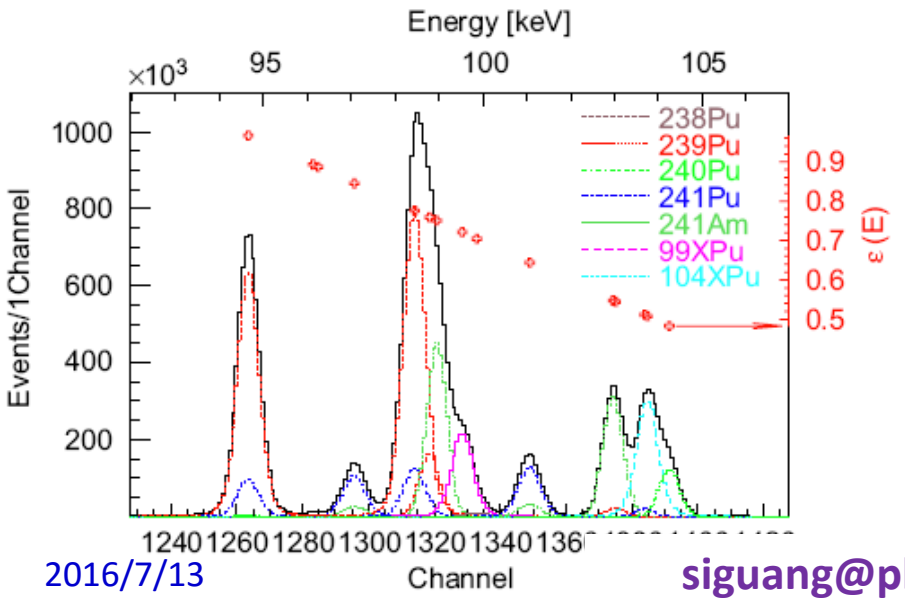
Nuclear Instruments and Methods in Physics Research A

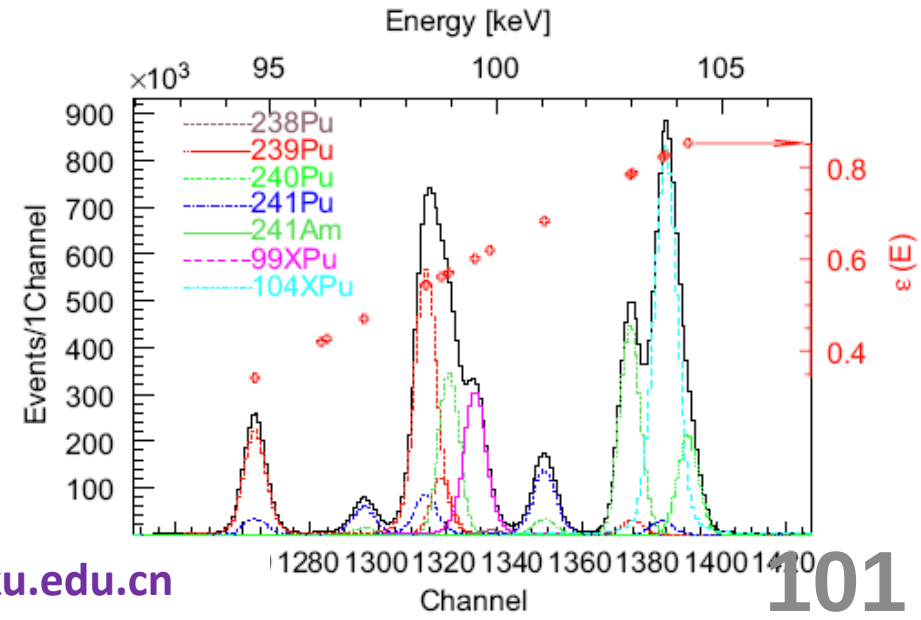
journal homepage: www.elsevier.com/locate/nima

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



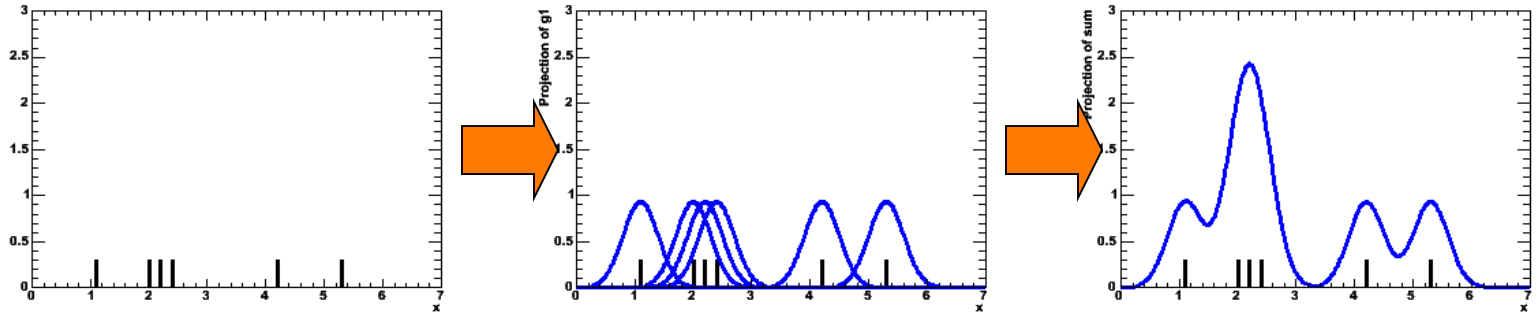


- Class `RooKeysPdf` – A kernel estimation p.d.f.
  - Uses *unbinned* data
  - Idea represent each event of your MC sample as a Gaussian probability distribution
  - Add probability distributions from all events in sample

Sample of events

Gaussian probability distributions for each event

Summed probability distribution for all events in sample

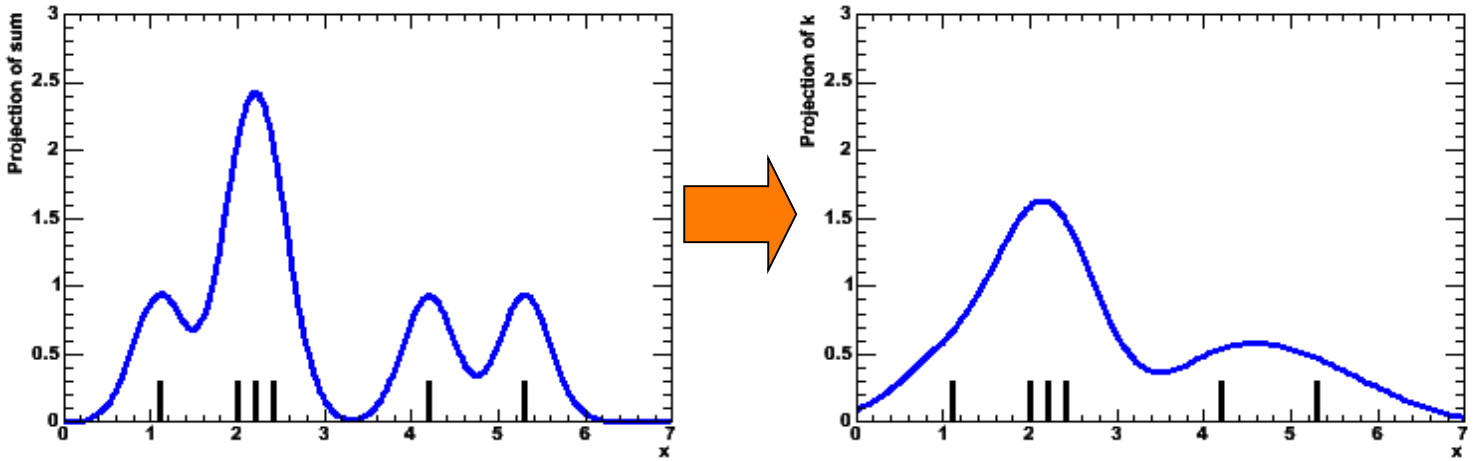


Highlight of non-parametric shapes – kernel estimation

- 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(Automatically calculated）
  - Choose **wide** Gaussian if local density of events is **low** -->Preserves small features in high statistics areas,
  - Choose **narrow** Gaussian if local density of events is high→ minimize jitter in low statistics areas

Static Kernel
(with of all Gaussian identical)

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



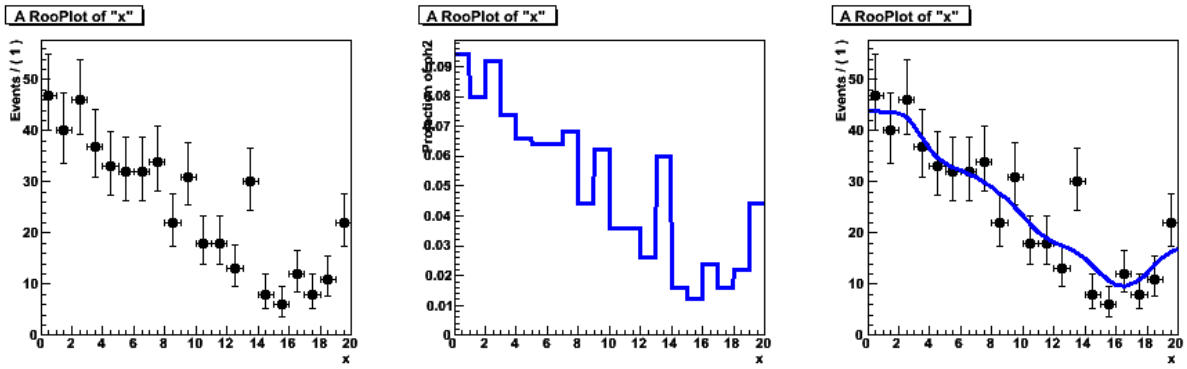
siguang@pku.edu.cn

Highlight of non-parametric shapes – kernel estimation

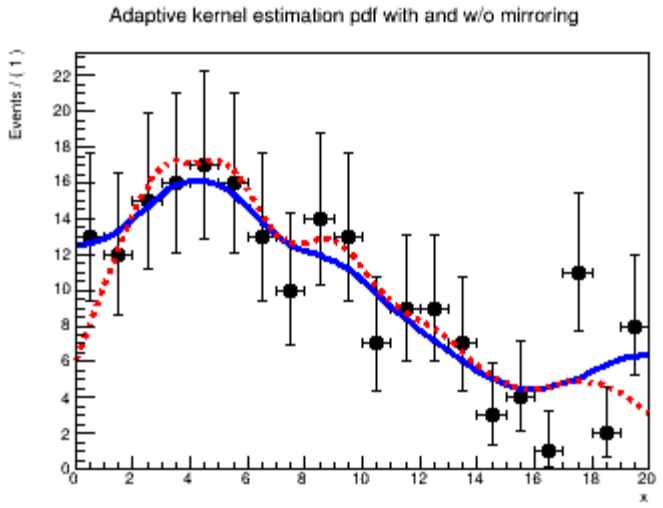
```
// Adaptive kernel estimation p.d.f
 RooKeysPdf k("k","k",x,*d,RooKeysPdf::MirrorBoth) ;
```

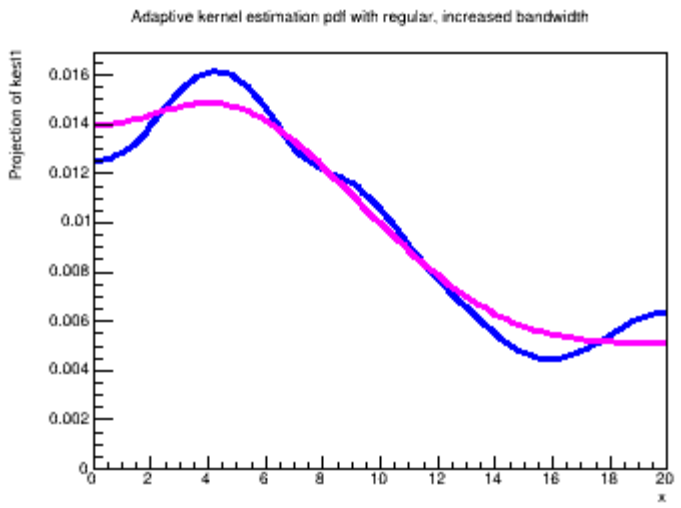
- Example with comparison to histogram based p.d.f
  - Superior performance at low statistics
  - Can mirror input data over boundaries to reduce 'edge leakage'
  - Works also in >1 dimensions (class **RooNDKeysPdf**)

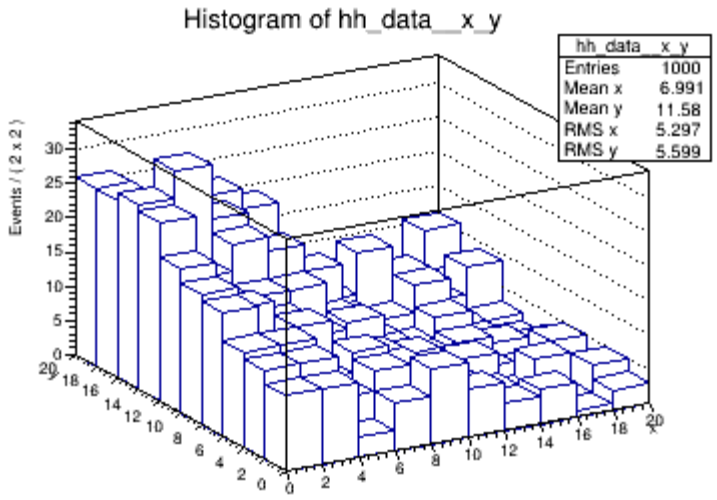
Data (N=500) RooHistPdf(data) RooKeysPdf(data)

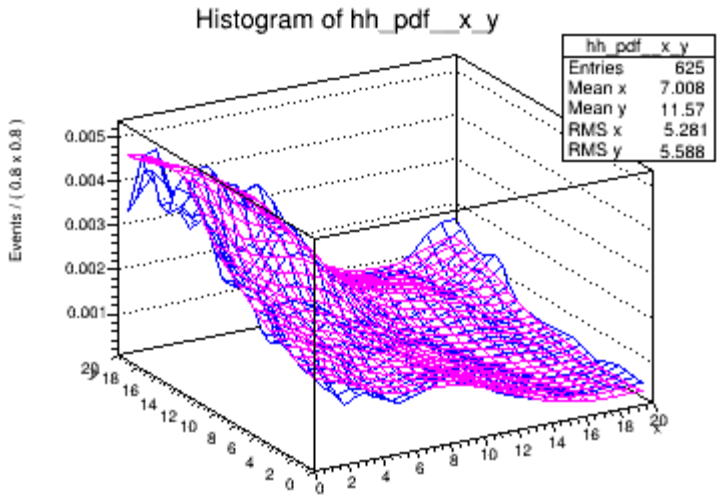


$ROOTSYS/tutorials/roofit/rf707_kernelestimation.C









siguang@pku.edu.cn

$ROOTSYS/tutorials/roofit/rf707_kernelestimation.C

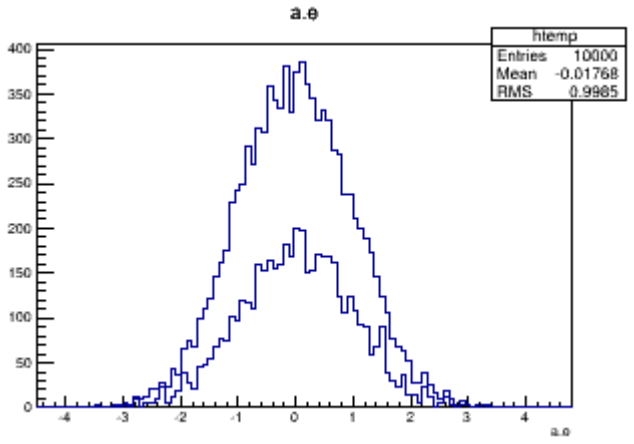
```
void rf707_kernelestimation()
{
  // C r e a t e   l o w   s t a t s   1 - D   d a t a s e t
  // ---------------------------------------------------

  // 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) ;
```

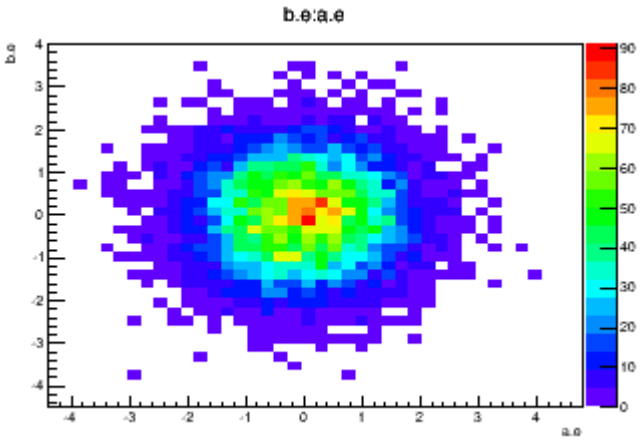
$ROOTSYS/tutorials/roofit/rf707_kernelestimation.C

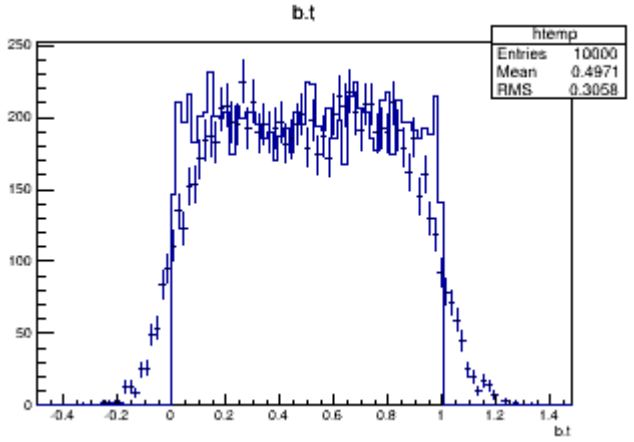
```
// Adaptive kernel estimation pdf with increased bandwidth scale factor
// (promotes smoothness over detail preservation)
RooKeysPdf kest3("kest1","kest1",x,*data1,RooKeysPdf::MirrorBoth,2) ;


// 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)) ;
```

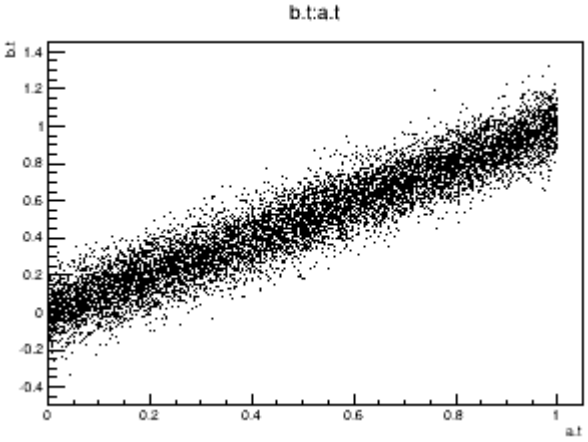
.....

$ROOTSYS/tutorials/tree/tree0.C









$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;
}
```

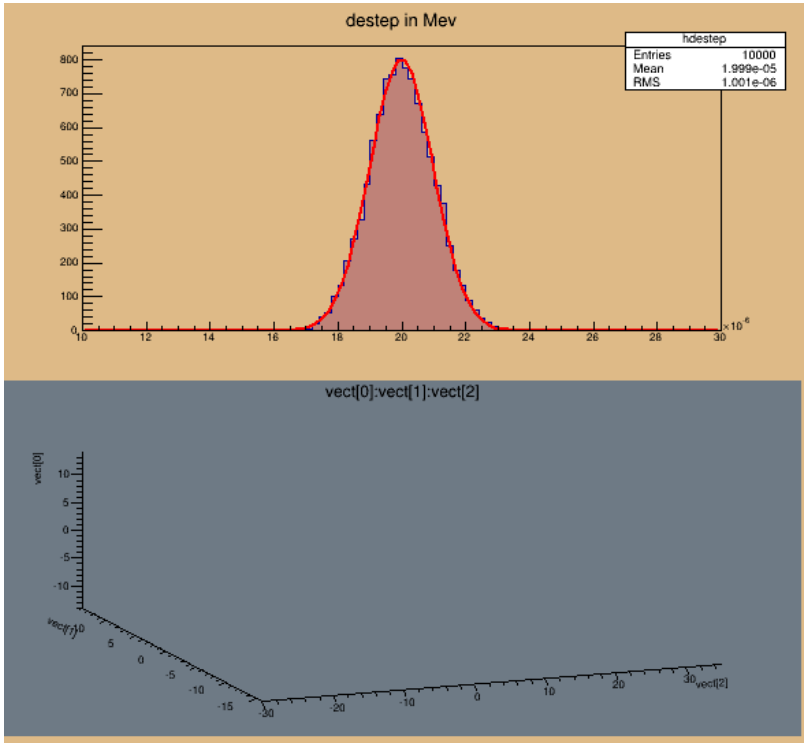
$ROOTSYS/tutorials/tree/tree1.C

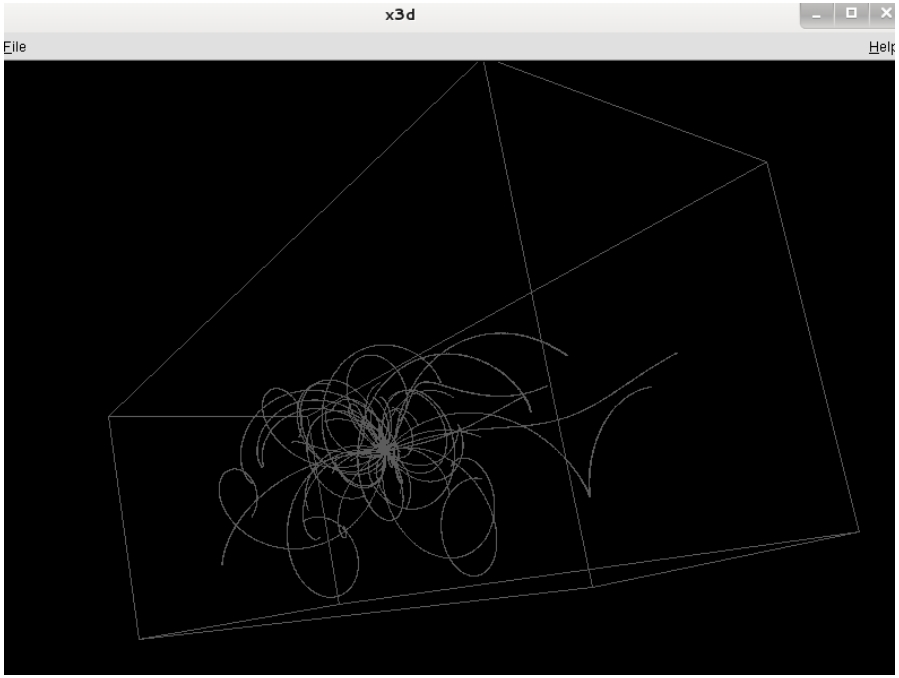
```
void tree1w()
{
   //create a Tree file tree1.root

   //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;
```

Branch(branchname, &object, bufsize, splitlevel)

$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;
      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();
}
```

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} )
```

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;
}
```


siguang@pku.edu.cn

可以把ROOT作为普通库函数进行连接

编译及运行

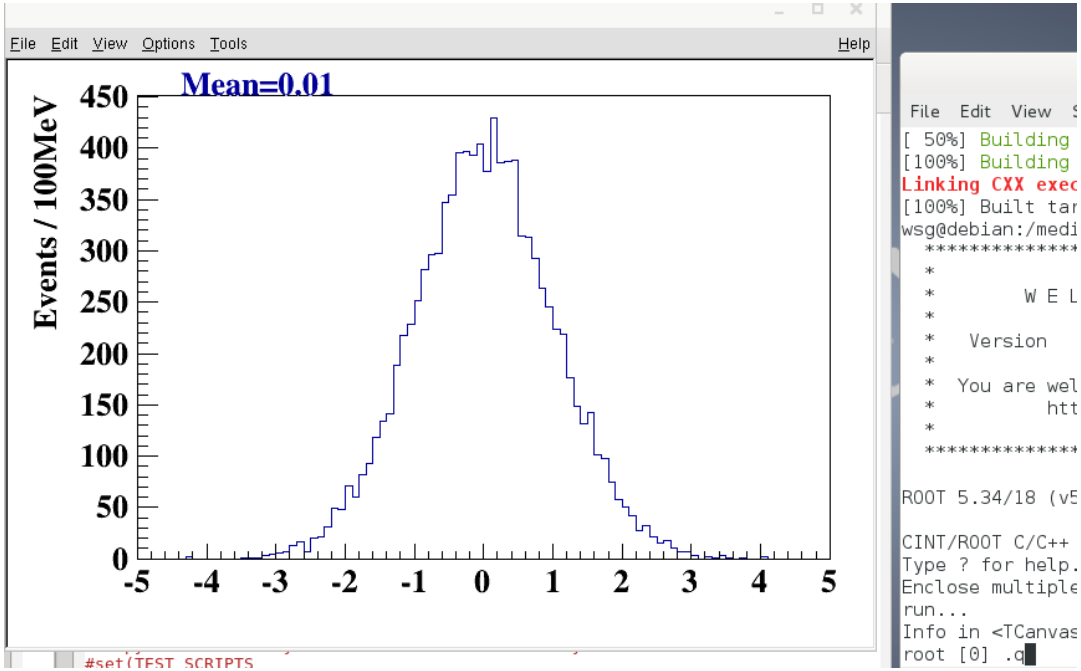
```
>cmake ../GuiRunCode/
>make
>./runTest
root[0].q
```

注：后者为CmakeLists.txt 所在的目录

注：生成可执行程序 runTest

注：运行

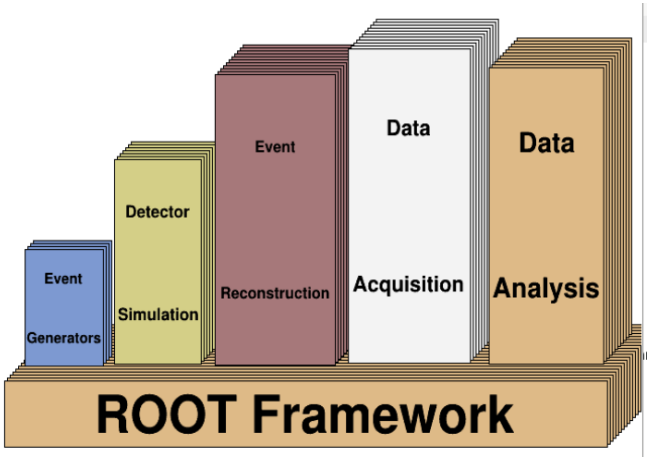
注： .q 退出



siguang@pku.edu.cn

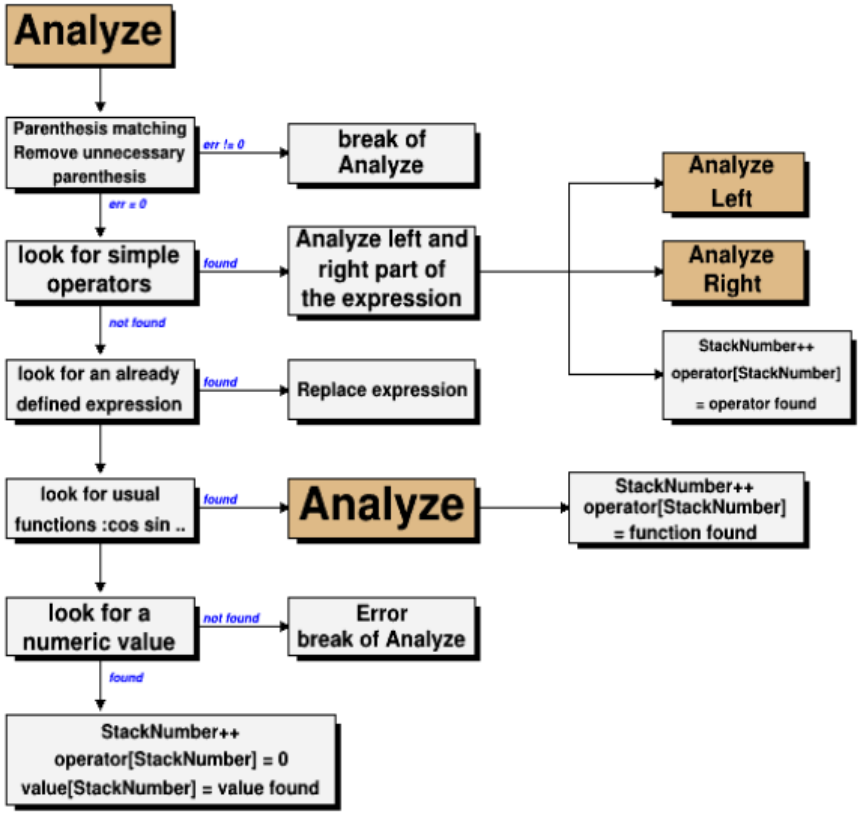
Codes under $ROOTSYS/tutorials/graphics

graphics/framework.C



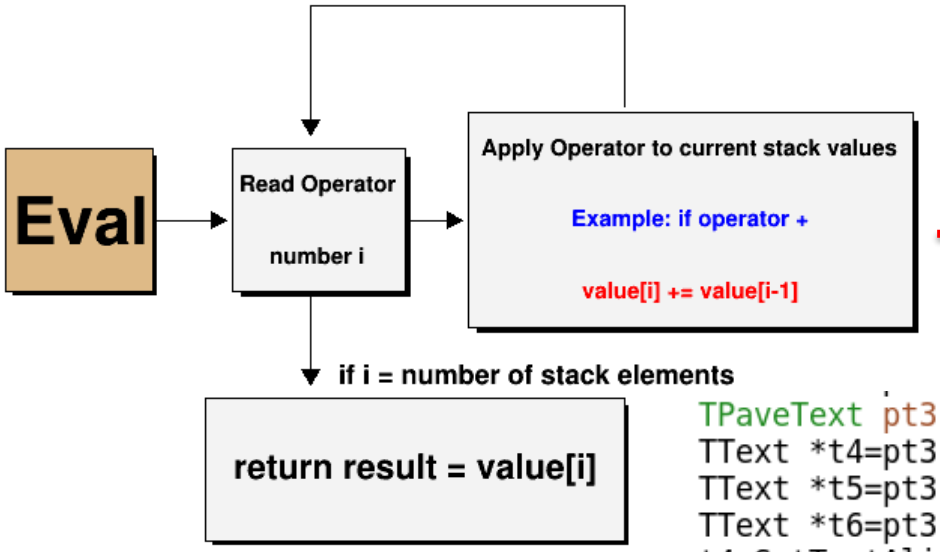

```
TCanvas *c1 = new TCanvas("c1","The ROOT Framework",200,10,700,500);
c1->Range(0,0,19,12);
//
TPavesText *rootf = new TPavesText(0.4,0.6,18,2.3,20,"tr");
rootf->AddText("ROOT Framework");
rootf->SetFillColor(42);
rootf->Draw();
//
TPavesText *eventg = new TPavesText(0.99,2.66,3.29,5.67,4,"tr");
eventg->SetFillColor(38);
eventg->AddText("Event");
eventg->AddText("Generators");
eventg->Draw();
//
TPavesText *simul = new TPavesText(3.62,2.71,6.15,7.96,7,"tr");
simul->SetFillColor(41);
simul->AddText("Detector");
simul->AddText("Simulation");
simul->Draw();
```

graphics/analyze.C



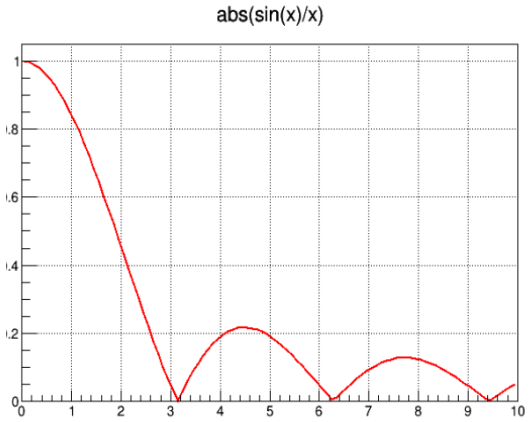

```
..
//This macro produces the flowchart of TFormula::Analyze
//Author: Rene Brun

gROOT->Reset();
c1 = new TCanvas("c1","Analyze.mac",620,790);
c1->Range(-1,0,19,30);
TPaveLabel pl1(0,27,3.5,29,"Analyze");
pl1.SetFillColor(42);
pl1.Draw();
TPaveText pt1(0,22.8,4,25.2);
TText *t1=pt1.AddText("Parenthesis matching");
TText *t2=pt1.AddText("Remove unnecessary");
TText *t2a=pt1.AddText("parenthesis");
pt1.Draw();
TPaveText pt2(6,23,10,25);
TText *t3=pt2.AddText("break of");
TText *t4=pt2.AddText("Analyze");
pt2.Draw();

 ......

TArrow ar(2,27,2,25.4,0.012,"|>");
ar.SetFillColor(1);
ar.Draw();
ar.DrawArrow(2,22.8,2,21.2,0.012,"|>");
ar.DrawArrow(2,19,2,17.2,0.012,"|>");
```

graphics/eval.C




```
TPaveText pt3(9,3.5,17.5,6.5);
TText *t4=pt3.AddText("Apply Operator to current stack values");
TText *t5=pt3.AddText("Example: if operator +");
TText *t6=pt3.AddText("value[i] += value[i-1]");
t4.SetTextAlign(22);
t5.SetTextAlign(22);
t6.SetTextAlign(22);
t5.SetTextColor(4);
t6.SetTextColor(2);
pt3.Draw();
```

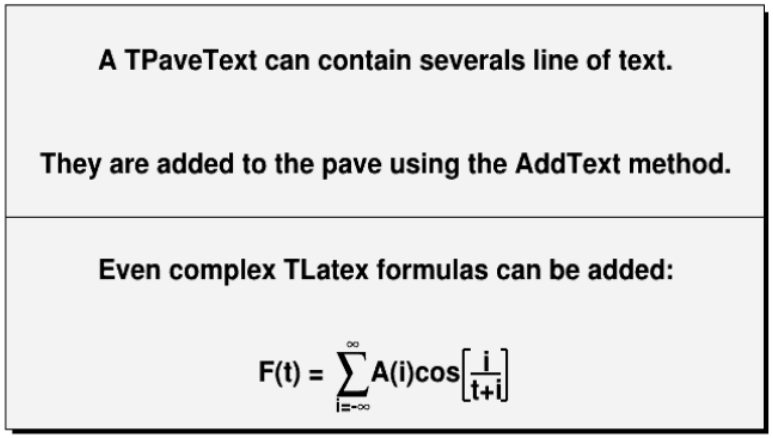
graphics/formula1.C



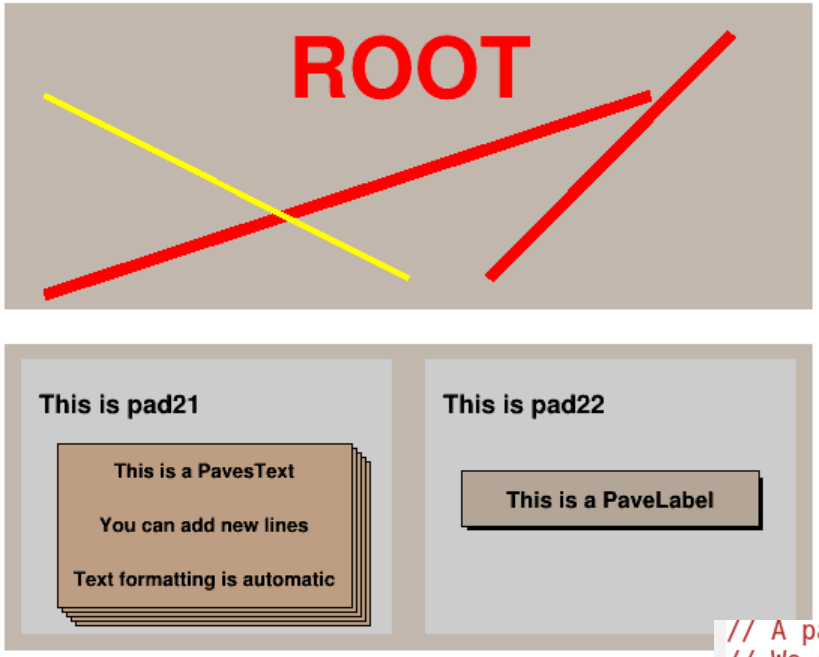

```
,,
form1 = new TFormula("form1","sqrt(abs(x))");
form1->Eval(2);
form1->Eval(-45);
//
// Create a one dimensional function and draw it
//
fun1 = new TF1("fun1","abs(sin(x)/x)",0,10);
c1->SetGridx();
c1->SetGridy();
fun1->Draw();
c1->Update();
//
// Before leaving this demo, we print the list of objects known to ROOT
//
if (gObjectTable) gObjectTable->Print();
```

/graphics/pavetext.C

```
{
   TCanvas *c = new TCanvas("c");
   TPaveText *pt = new TPaveText(.05,.1,.95,.8);

   pt->AddText("A TPaveText can contain severals
   pt->AddText("They are added to the pave using the AddText method.");
   pt->AddLine(.0,.5,1.,.5);
   pt->AddText("Even complex TLatex formulas can be added:");
   pt->AddText("F(t) = #sum_{i=-#infty}^{#infty}A(i)cos#[]{#frac{i}{t+i}}");

   pt->Draw();
   return c;
}
```



graphics/canvas.C




```
  c1 = new TCanvas("c1","Canvas Example",200,10,600,480);

  gBenchmark->Start("canvas");
//
// Inside this canvas, we create two pads
//
  pad1 = new TPad("pad1","This is pad1",0.05,0.52,0.95,0.97);
  pad2 = new TPad("pad2","This is pad2",0.05,0.02,0.95,0.47);
  pad1->SetFillColor(11);
  pad2->SetFillColor(11);
  pad1->Draw();
  pad2->Draw();
```

```
// A pad may contain other pads and graphics objects.
// We set the current pad to pad2.
// Note that the current pad is always highlighted.
//
  pad2->cd();
  pad21 = new TPad("pad21","First subpad of pad2",0.02,0.05,0.48,0.95,17,3);
  pad22 = new TPad("pad22","Second subpad of pad2",0.52,0.05,0.98,0.95,17,3);
  pad21->Draw();
  pad22->Draw();
```

```
//
// We enter some primitives in the created pads and set some attributes
//
  pad1->cd();
  float xt1 = 0.5;
  float yt1 = 0.1;
  t1 = new TText(0.5,yt1,"ROOT");
  t1->SetTextAlign(22);
  t1->SetTextSize(0.05);
  t1->Draw();
  line1 = new TLine(0.05,0.05,0.80,0.70);
  line1->SetLineWidth(8);
  line1->SetLineColor(2);
  line1->Draw();
  line1->DrawLine(0.6,0.1,0.9,0.9);
  line2 = new TLine(0.05,0.70,0.50,0.10);
  line2->SetLineWidth(4);
  line2->SetLineColor(5);
  line2->Draw();
```

```
for (int i=0;i<nloops;i++) {
   color++;
   color %= 8;
   line1->SetLineColor(color);
   t1->SetTextSize(t10 + t1ds*i);
   t1->SetTextColor(color);
   t1->SetX(xt1+dxt1*i);
   t1->SetY(yt1+dyt1*i);
   pad1->Modified();
   paves->SetX2NDC(xp2+dxp2*i);
   paves->SetY2NDC(yp2+dyp2*i);
   pad21->Modified();
   label->SetX1NDC(xlc+dxlc*i);
   label->SetY1NDC(ylc+dylc*i);
   label->SetX2NDC(xlc+dxlc*i+0.8);
   label->SetY2NDC(ylc+dylc*i+0.2);
   pad22->Modified();
   c1->Update();
}
```

The text will move after re-draw

/tutorials/graphics/first.C

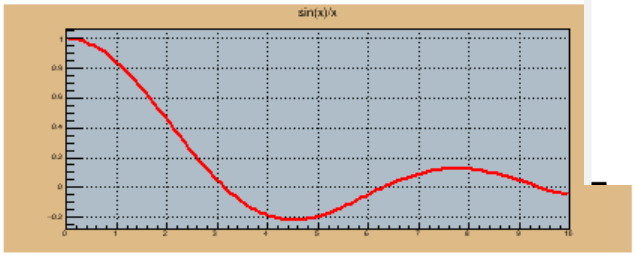
My first ROOT interactive session

ROOT is based on CINT, a powerful C/C++ interpreter.

Blocks of lines can be entered within {...}.

Previous typed lines can be recalled.

```
Root > float x=5; float y=7;
Root > x*sqrt(y)
(double)1.322875655532e+01
Root > for (int i=2;i<7;i++) printf("sqrt(%d) = %f",i,sqrt(i));
sqrt(2) = 1.414214
sqrt(3) = 1.732051
sqrt(4) = 2.000000
sqrt(5) = 2.236068
sqrt(6) = 2.449490
Root > TF1 f1("f1","sin(x)/x",0,10)
Root > f1.Draw()
```

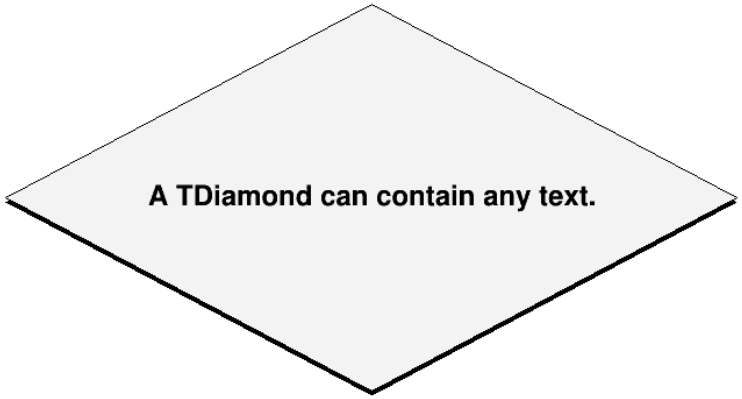



```
//Show some basic primitives
//Author: Rene Brun
void first() {

   TCanvas *nut = new TCanvas("nut", "FirstSession",100,10,700,900);
   nut->Range(0,0,20,24);
   nut->SetFillColor(10);
   nut->SetBorderSize(2);

   TPaveLabel *pl = new TPaveLabel(3,22,17,23.7,
      "My first ROOT interactive session","br");
   pl->SetFillColor(18);
   pl->Draw();

   TText t(0,0,"a");
   t.SetTextFont(62);
   t.SetTextSize(0.025);
   t.SetTextAlign(12);
   t.DrawText(2,20.3,"ROOT is based on CINT, a powerful C/C++ interpreter.");
   t.DrawText(2,19.3,"Blocks of lines can be entered within {...}.");
   t.DrawText(2,18.3,"Previous typed lines can be recalled.");
```

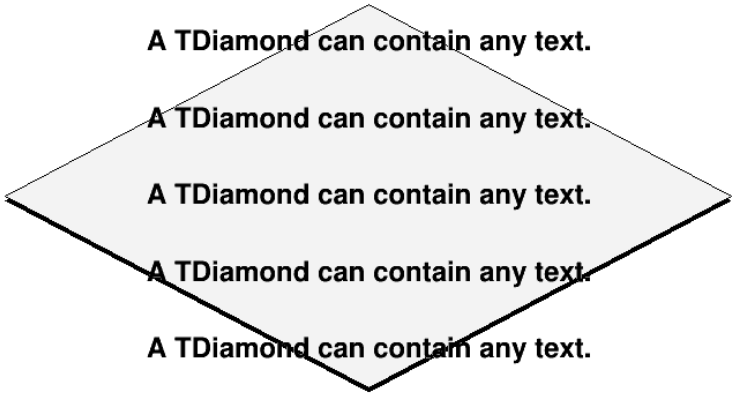
```
t.SetTextFont(72);
t.SetTextSize(0.026);
t.DrawText(3,17,"Root >  float x=5; float y=7;");
t.DrawText(3,16,"Root >  x*sqrt(y)");
t.DrawText(3,14,
   "Root >  for (int i=2;i<7;i++) printf(\"sqrt(%d) = %f\",i,sqrt(i));");
t.DrawText(3,10,"Root >  TF1 f1(\"f1\",\"sin(x)/x\",0,10)");
t.DrawText(3, 9,"Root >  f1.Draw()");
t.SetTextFont(81);
t.SetTextSize(0.018);
t.DrawText(4,15,"(double)1.322875655532e+01");
t.DrawText(4,13.3,"sqrt(2) = 1.414214");
t.DrawText(4,12.7,"sqrt(3) = 1.732051");
t.DrawText(4,12.1,"sqrt(4) = 2.000000");
t.DrawText(4,11.5,"sqrt(5) = 2.236068");
t.DrawText(4,10.9,"sqrt(6) = 2.449490");
```

```
    TPad *pad = new TPad("pad","pad",.2,.05,.8,.35);
    pad->SetFillColor(42);
    pad->SetFrameFillColor(33);
    pad->SetBorderSize(10);
    pad->Draw();
    pad->cd();
    pad->SetGrid();
    TF1 *f1 = new TF1("f1","sin(x)/x",0,10);
    f1->Draw();
    nut->cd();
}
```

graphics/diamond.C

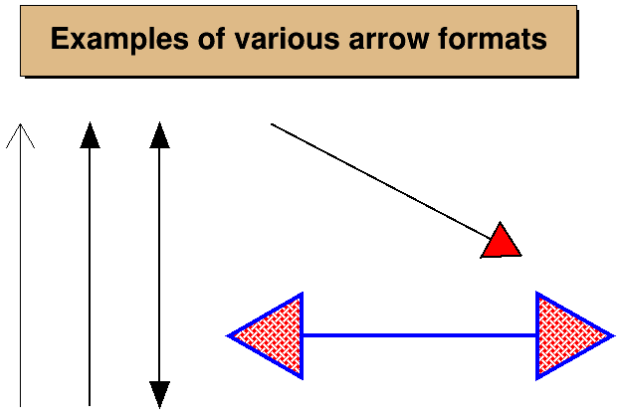



```
{
   TCanvas *c = new TCanvas("c");
   TDiamond *d = new TDiamond(.05,.1,.95,.8);

   d->AddText("A TDiamond can contain any text.");

   d->Draw();
   return c;
}
```



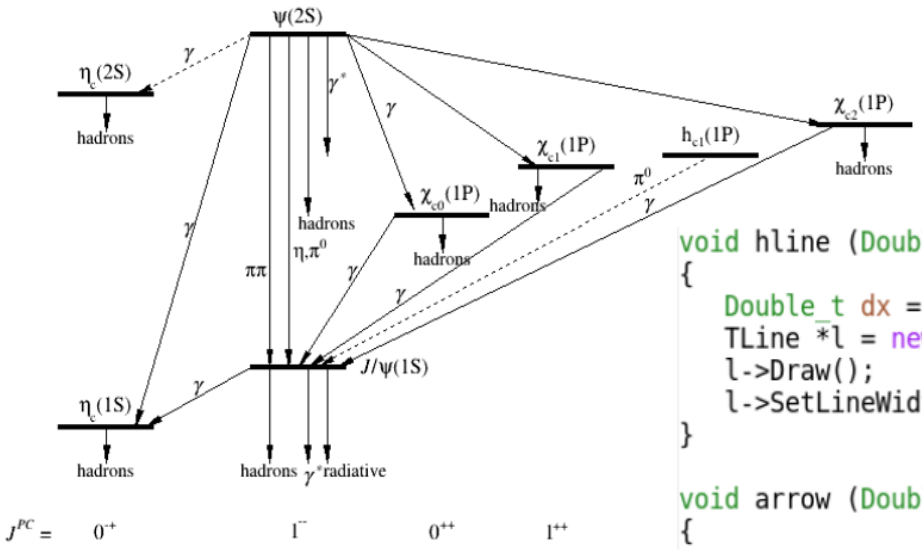

```
{
   TCanvas *c = new TCanvas("c");
   TDiamond *d = new TDiamond(.05,.1,.95,.8);

   d->AddText("A TDiamond can contain any text.");
   d->AddText("A TDiamond can contain any text.");
   d->AddText("A TDiamond can contain any text.");
   d->AddText("A TDiamond can contain any text.");
   d->AddText("A TDiamond can contain any text.");

   d->Draw();
   return c;
}
```

graphics/arrow.C




```
c1 = new TCanvas("c1");
c1->Range(0,0,1,1);
TPaveLabel par(0.1,0.8,0.9,0.95,"Examples of various arrow formats");
par.SetFillColor(42);
par.Draw();
TArrow ar1(0.1,0.1,0.1,0.7);
ar1.Draw();
TArrow ar2(0.2,0.1,0.2,0.7,0.05,"|>");
ar2.SetAngle(40);
ar2.SetLineWidth(2);
ar2.Draw();
TArrow ar3(0.3,0.1,0.3,0.7,0.05,"<|>");
ar3.SetAngle(40);
ar3.SetLineWidth(2);
ar3.Draw();
TArrow ar4(0.46,0.7,0.82,0.42,0.07,"|>");
ar4.SetAngle(60);
ar4.SetLineWidth(2);
ar4.SetFillColor(2);
ar4.Draw();
TArrow ar5(0.4,0.25,0.95,0.25,0.15,"<|>");
ar5.SetAngle(60);
ar5.SetLineWidth(4);
ar5.SetLineColor(4);
ar5.SetFillStyle(3008);
ar5.SetFillColor(2);
ar5.Draw();
return c1;
```

graphics/mass_spectrum.C



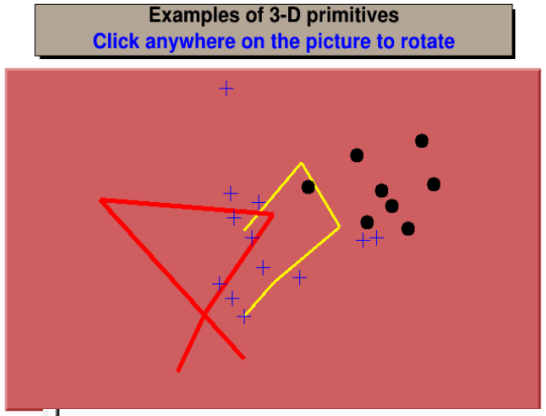

```
void hline (Double_t x, Double_t y)
{
   Double_t dx = 0.1;
   TLine *l = new TLine(x,y,x+dx,y);
   l->Draw();
   l->SetLineWidth(4);
}

void arrow (Double_t x1, Double_t y1, Double_t x2, Double_t y2, Int_t ls)
{
   arrow = new TArrow(x1,y1,x2,y2,0.025,"|>");
   arrow->SetFillColor(1);
   arrow->SetFillStyle(1001);
   arrow->SetLineStyle(ls);
   arrow->SetAngle(19);
   arrow->Draw();
}
```

```
void mass_spectrum()
{
   TCanvas *C = new TCanvas("C","C",800,500);

   hline (0.10,0.25);
   hline (0.10,0.80);
   hline (0.30,0.90);
   hline (0.30,0.35);
   hline (0.45,0.60);
   hline (0.58,0.68);
   hline (0.73,0.70);
   hline (0.89,0.75);

   arrow(0.32, 0.90, 0.32, 0.35, 1);
   arrow(0.34, 0.90, 0.34, 0.35, 1);
   arrow(0.36, 0.90, 0.36, 0.60, 1);
   arrow(0.38, 0.90, 0.38, 0.70, 1);
```

……..

```
TLatex l2;
l2.SetTextSize(0.038);
l2.SetTextAlign(22);
l2.SetTextFont(132);
l2.DrawLatex(0.07, 0.08, "#font[12]{J^{PC}} =");
l2.DrawLatex(0.15, 0.08, "0^{-+}");
l2.DrawLatex(0.35, 0.08, "1^{--}");
l2.DrawLatex(0.50, 0.08, "0^{++}");
l2.DrawLatex(0.62, 0.08, "1^{++}");
l2.DrawLatex(0.77, 0.08, "1^{+-}");
l2.DrawLatex(0.93, 0.08, "2^{++}");
l2.DrawLatex(0.15, 0.83, "#eta_{c}(2S)");
l2.DrawLatex(0.15, 0.28, "#eta_{c}(1S)");
```

graphics/basic3d.C




```
{
   // Show 3-D polylines and markers
   // To see the output of this macro, click begin_html <a href="gif/basic3d.gif
">here</a> end_html
   //
   gROOT->Reset();
   c1 = new TCanvas("c1","PolyLine3D & PolyMarker3D Window",200,10,700,500);

   // create a pad
   p1 = new TPad("p1","p1",0.05,0.02,0.95,0.82,46,3,1);
   p1->Draw();
   p1->cd();

   // creating a view
   view = TView::CreateView(1);
   view->SetRange(5,5,5,25,25,25);
```

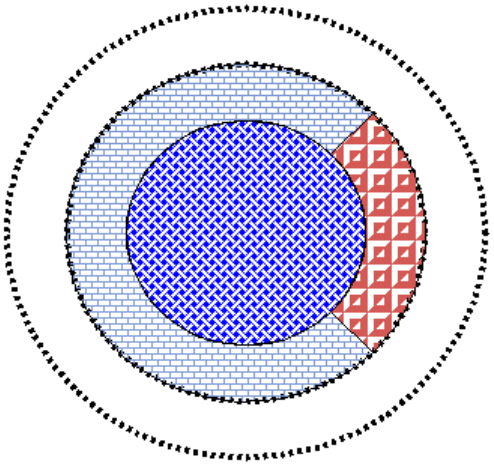
SetRange(Double_t x0, Double_t y0, Double_t z0, Double_t x1, Double_t y1, Double_t z1, Int_t flag = 0)

```
// create a first PolyLine3D
TPolyLine3D *pl3d1 = new TPolyLine3D(5);

// set points
pl3d1->SetPoint(0, 10, 10, 10);
pl3d1->SetPoint(1, 15, 15, 10);
pl3d1->SetPoint(2, 20, 15, 15);
pl3d1->SetPoint(3, 20, 20, 20);
pl3d1->SetPoint(4, 10, 10, 20);
// set attributes
pl3d1->SetLineWidth(3);
pl3d1->SetLineColor(5);
```

```
TPolyMarker3D *pm3d1 = new TPolyMarker3D(12);

// set points
pm3d1->SetPoint(0, 10, 10, 10);
pm3d1->SetPoint(1, 11, 15, 11);
pm3d1->SetPoint(2, 12, 15, 9);
pm3d1->SetPoint(3, 13, 17, 20);
pm3d1->SetPoint(4, 14, 16, 15);
pm3d1->SetPoint(5, 15, 20, 15);
pm3d1->SetPoint(6, 16, 18, 10);
pm3d1->SetPoint(7, 17, 15, 10);
pm3d1->SetPoint(8, 18, 22, 15);
pm3d1->SetPoint(9, 19, 28, 25);
pm3d1->SetPoint(10, 20, 12, 15);
pm3d1->SetPoint(11, 21, 12, 15);

// set marker size, color & style
pm3d1->SetMarkerSize(2);
pm3d1->SetMarkerColor(4);
pm3d1->SetMarkerStyle(2);
```

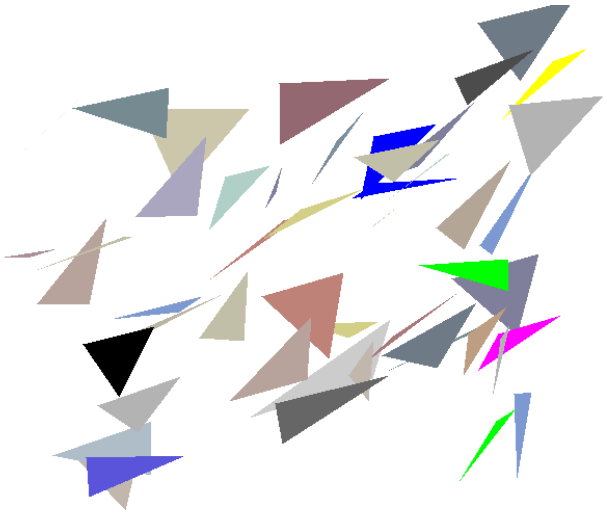
```
// draw
pl3d1->Draw();
pl3d2->Draw();
pm3d1->Draw();
pm3d2->Draw();
//
// draw a title/explanation in the canvas pad
c1->cd();
TPaveText *title = new TPaveText(0.1,0.85,0.9,0.97);
title->SetFillColor(24);
title->AddText("Examples of 3-D primitives");
TText *click=title->AddText("Click anywhere on the picture to rotate");
click->SetTextColor(4);
title->Draw();
```

graphics/crown.C



```
{
   TCanvas *c1 = new TCanvas("c1","c1",400,400);
   TCrown *cr1 = new TCrown(.5,.5,.3,.4);
   cr1->SetLineStyle(2);
   cr1->SetLineWidth(4);
   cr1->Draw();
   TCrown *cr2 = new TCrown(.5,.5,.2,.3,45,315);
   cr2->SetFillColor(38);
   cr2->SetFillStyle(3010);
   cr2->Draw();
   TCrown *cr3 = new TCrown(.5,.5,.2,.3,-45,45);
   cr3->SetFillColor(50);
   cr3->SetFillStyle(3025);
   cr3->Draw();
   TCrown *cr4 = new TCrown(.5,.5,.0,.2);
   cr4->SetFillColor(4);
   cr4->SetFillStyle(3008);
   cr4->Draw();
   return c1;
}
```

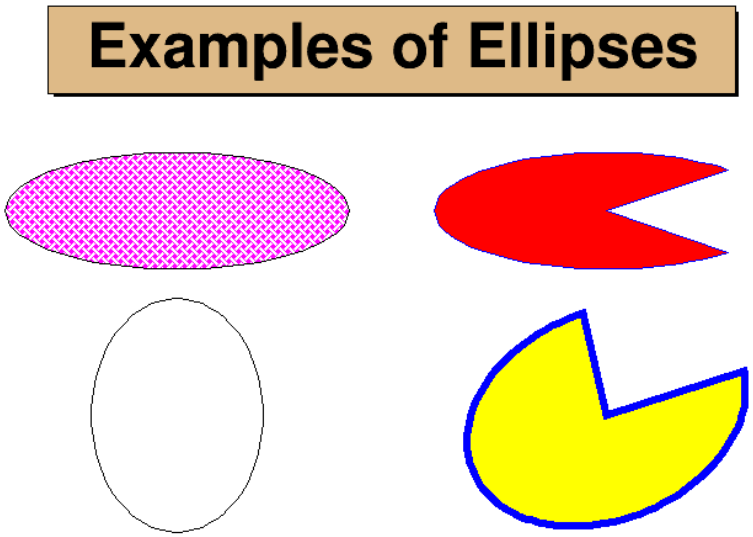
graphics/triangles.C



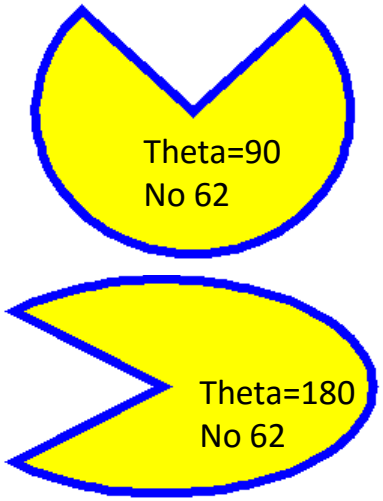
```
TCanvas *c1 = new TCanvas("c1","triangles",10,10,700,700);
TRandom r;
Double_t dx = 0.2; Double_t dy = 0.2;
Int_t ncolors = gStyle->GetNumberOfColors();
Double_t x[4],y[4];
TColor *c;
Int_t ci;
for (Int_t i=0;i<ntriangles;i++) {
   x[0] = r.Uniform(.05,.95); y[0] = r.Uniform(.05,.95);
   x[1] = x[0] + dx*r.Rndm(); y[1] = y[0] + dy*r.Rndm();
   x[2] = x[1] - dx*r.Rndm(); y[2] = y[1] - dy*r.Rndm();
   x[3] = x[0];               y[3] = y[0];
   TPolyLine *pl = new TPolyLine(4,x,y);
   pl->SetUniqueID(i);
   ci = ncolors*r.Rndm();
   c  = gROOT->GetColor(ci);
   c->SetAlpha(r.Rndm());
   pl->SetFillColor(ci);
   pl->Draw("f");
}
c1->AddExec("ex","TriangleClicked()");
```

```
void TriangleClicked() {
   //this action function is called whenever you move the mouse
   //it just prints the id of the picked triangle
   //you can add graphics actions instead
   int event = gPad->GetEvent();
   if (event != 11) return; //may be comment this line
   TObject *select = gPad->GetSelected();
   if (!select) return;
   if (select->InheritsFrom(TPolyLine::Class())) {
      TPolyLine *pl = (TPolyLine*)select;
      printf("You have clicked triangle %d, color=%d\n",
             pl->GetUniqueID(),pl->GetFillColor());
   }
}
```

graphics/ellipse.C



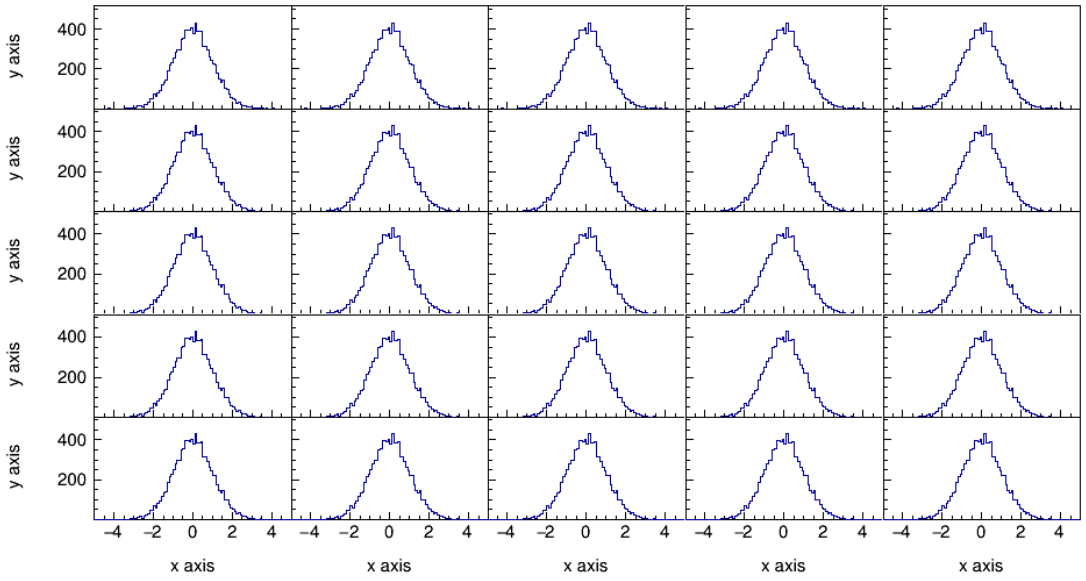

```
c1 = new TCanvas("c1");
c1->Range(0,0,1,1);
TPaveLabel pel(0.1,0.8,0.9,0.95,"Examples of Ellipses");
pel.SetFillColor(42);
pel.Draw();
TEllipse el1(0.25,0.25,.1,.2);
el1.Draw();
TEllipse el2(0.25,0.6,.2,.1);
el2.SetFillColor(6);
el2.SetFillStyle(3008);
el2.Draw();
TEllipse el3(0.75,0.6,.2,.1,45,315);
el3.SetFillColor(2);
el3.SetFillStyle(1001);
el3.SetLineColor(4);
el3.Draw();
TEllipse el4(0.75,0.25,.2,.15,45,315,62)
el4.SetFillColor(5);
el4.SetFillStyle(1001);
el4.SetLineColor(4);
el4.SetLineWidth(6);
el4.Draw();
```



virtual void DrawEllipse(Double_t x1, Double_t y1, Double_t r1, Double_t r2, Double_t phimin, Double_t phimax, Double_t theta, Option_t* option = "")

graphics/canvas2.C

//Example of canvas partitioning
// Sometimes the Divide() method is not appropriate to divide a Canvas. Because of the left and right margins, all the pads do not have the same width and height. CanvasPartition does that properly. This example also ensure that the axis labels and titles have the same sizes and that the tick marks length is uniform.



siguang@pku.edu.cn

```
TCanvas *C = (TCanvas*) gROOT->FindObject("C");
if (C) delete C;
C = new TCanvas("C","canvas",1024,640);
C->SetFillStyle(4000);

// Number of PADS
const Int_t Nx = 5;
const Int_t Ny = 5;

// Margins
Float_t lMargin = 0.12;
Float_t rMargin = 0.05;
Float_t bMargin = 0.15;
Float_t tMargin = 0.05;

// Canvas setup
CanvasPartition(C,Nx,Ny,lMargin,rMargin,bMargin,tMargin);
```

```
// Dummy histogram.
TH1F *h = (TH1F*) gROOT->FindObject("histo");
if (h) delete h;
h = new TH1F("histo","",100,-5.0,5.0);
h->FillRandom("gaus",10000);
h->GetXaxis()->SetTitle("x axis");
h->GetYaxis()->SetTitle("y axis");

TPad *pad[Nx][Ny];

for (Int_t i=0;i<Nx;i++) {
   for (Int_t j=0;j<Ny;j++) {
      C->cd(0);

      // Get the pads previosly created.
      char pname[16];
      sprintf(pname,"pad_%i_%i",i,j);
      pad[i][j] = (TPad*) gROOT->FindObject(pname);
      pad[i][j]->Draw();
      pad[i][j]->SetFillStyle(4000);
      pad[i][j]->SetFrameFillStyle(4000);
      pad[i][j]->cd();
```

```
// Size factors
Float_t xFactor = pad[0][0]->GetAbsWNDC()/pad[i][j]->GetAbsWNDC();
Float_t yFactor = pad[0][0]->GetAbsHNDC()/pad[i][j]->GetAbsHNDC();

char hname[16];
sprintf(hname,"h_%i_%i",i,j);
TH1F *hFrame = (TH1F*) h->Clone(hname);
hFrame->Reset();
hFrame->Draw();

// y axis range
hFrame->GetYaxis()->SetRangeUser(0.0001,1.2*h->GetMaximum());
```

Remove 0 in Y axis

```
// Format for y axis
hFrame->GetYaxis()->SetLabelFont(43);
hFrame->GetYaxis()->SetLabelSize(16);
hFrame->GetYaxis()->SetLabelOffset(0.02);
hFrame->GetYaxis()->SetTitleFont(43);
hFrame->GetYaxis()->SetTitleSize(16);
hFrame->GetYaxis()->SetTitleOffset(5);

hFrame->GetYaxis()->CenterTitle();
hFrame->GetYaxis()->SetNdivisions(505);

// TICKS Y Axis
hFrame->GetYaxis()->SetTickLength(xFactor*0.04/yFactor);

// Format for x axis
hFrame->GetXaxis()->SetLabelFont(43);
hFrame->GetYaxis()->SetLabelSize(16);
  ……
// TICKS X Axis
hFrame->GetXaxis()->SetTickLength(yFactor*0.06/xFactor);

h->Draw("same");
```

```
void CanvasPartition(TCanvas *C,const Int_t Nx = 2,const Int_t Ny = 2,
                     Float_t lMargin = 0.15, Float_t rMargin = 0.05,
                     Float_t bMargin = 0.15, Float_t tMargin = 0.05)
{
   if (!C) return;

   // Setup Pad layout:
   Float_t vSpacing = 0.0;
   Float_t vStep  = (1.- bMargin - tMargin - (Ny-1) * vSpacing) / Ny;

   Float_t hSpacing = 0.0;
   Float_t hStep  = (1.- lMargin - rMargin - (Nx-1) * hSpacing) / Nx;

   Float_t vposd,vposu,vmard,vmaru,vfactor;
   Float_t hposl,hposr,hmarl,hmarr,hfactor;

   for (Int_t i=0;i<Nx;i++) {
```

```
if (i==0) {
    hposl = 0.0;
    hposr = lMargin + hStep;
    hfactor = hposr-hposl;
    hmarl = lMargin / hfactor;
    hmarr = 0.0;
} else if (i == Nx-1) {
    hposl = hposr + hSpacing;
    hposr = hposl + hStep + rMargin;
    hfactor = hposr-hposl;
    hmarl = 0.0;
    hmarr = rMargin / (hposr-hposl);
} else {
    hposl = hposr + hSpacing;
    hposr = hposl + hStep;
    hfactor = hposr-hposl;
    hmarl = 0.0;
    hmarr = 0.0;
}
```

```
for (Int_t j=0;j<Ny;j++) {

   if (j==0) {
      vposd = 0.0;
      vposu = bMargin + vStep;
      vfactor = vposu-vposd;
      vmard = bMargin / vfactor;
      vmaru = 0.0;
   } else if (j == Ny-1) {
      vposd = vposu + vSpacing;
      vposu = vposd + vStep + tMargin;
      vfactor = vposu-vposd;
      vmard = 0.0;
      vmaru = tMargin / (vposu-vposd);
   } else {
      vposd = vposu + vSpacing;
      vposu = vposd + vStep;
      vfactor = vposu-vposd;
      vmard = 0.0;
      vmaru = 0.0;
   }
```

```
            C->cd(0);

            char name[16];
            sprintf(name,"pad_%i_%i",i,j);
            TPad *pad = (TPad*) gROOT->FindObject(name);
            if (pad) delete pad;
            pad = new TPad(name,"",hposl,vposd,hposr,vposu);
            pad->SetLeftMargin(hmarl);
            pad->SetRightMargin(hmarr);
            pad->SetBottomMargin(vmard);
            pad->SetTopMargin(vmaru);

            pad->SetFrameBorderMode(0);
            pad->SetBorderMode(0);
            pad->SetBorderSize(0);

            pad->Draw();
        }
    }
}
```

graphics/latex.C

1) $C(x) = d\sqrt{\frac{2}{\lambda D}} \int_0^x \cos(\frac{\pi}{2}t^2)dt$

2) $C(x) = d\sqrt{\frac{2}{\lambda D}} \int^x \cos(\frac{\pi}{2}t^2)dt$

3) $R = |A|^2 = \frac{1}{2}\left(\left[\frac{1}{2}+ C(V)\right]^2+\left[\frac{1}{2}+S(V\right.$

4) $F(t) = \sum_{i=-\infty}^{\infty} A(i)\cos\left[\frac{i}{t+i}\right]$

5) ${}^{7}_{3}Li$

```
void latex() {
   TCanvas *c1 = new TCanvas("c1","test",600,700);
   // write formulas
   TLatex l;
   l.SetTextAlign(12);
   l.SetTextSize(0.04);
   l.DrawLatex(0.1,0.9,"1)   C(x) = d #sqrt{#frac{2}{#lambdaD}}\
   #int^{x}_{0}cos(#frac{#pi}{2}t^{2})dt");
   l.DrawLatex(0.1,0.7,"2)   C(x) = d #sqrt{#frac{2}{#lambdaD}}\
   #int^{x}cos(#frac{#pi}{2}t^{2})dt");
   l.DrawLatex(0.1,0.5,"3)   R = |A|^{2} = #frac{1}{2}#left(#[]{#fr
   C(V)}^{2}+#[]{#frac{1}{2}+S(V)}^{2}#right)");
   l.DrawLatex(0.1,0.3,
   "4)   F(t) = #sum_{i=-#infty}^{#infty}A(i)cos#[]{#frac{i}{t+i}}"
   l.DrawLatex(0.1,0.1,"5)   {}_{3}^{7}Li");
   c1->Print("latex.ps");
```

graphics/latex2.C

$e^+e^- \to Z^0 \to l\bar{l},\ q\bar{q}$

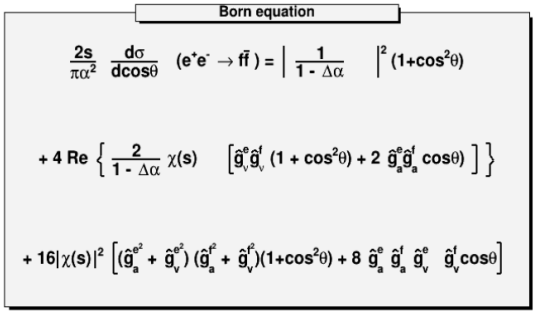
$|\vec{a}\bullet\vec{b}|=\Sigma a^i_{jk}+b^{bj}_i$

$i(\partial_\mu\bar{\psi}\gamma^\mu+m\bar{\psi})=0 \quad \Leftrightarrow(\Box+m^2)\psi=0$

$L_{em}=eJ^\mu_{em}A_\mu\ ,\ J^\mu_{em}=\bar{I}\quad \gamma_\mu I\ ,\ M^j_i=\Sigma A_\alpha\tau^{\alpha j}_i$

```
   TCanvas *c1 = new TCanvas("c1");
   TLatex l;
   l.SetTextAlign(23);
   l.SetTextSize(0.1);
   l.DrawLatex(0.5,0.95,"e^{+}e^{-}#rightarrowZ^{0}#rightarrowI#bar{I}, q#bar{q}
");
   l.DrawLatex(0.5,0.75,"|#vec{a}#bullet#vec{b}|=#Sigmaa^{i}_{jk}+b^{bj}_{i}");
   l.DrawLatex(0.5,0.5,"i(#partial_{#mu}#bar{#psi}#gamma^{#mu}+m#bar{#psi})=0\
   #Leftrightarrow(#Box+m^{2})#psi=0");
   l.DrawLatex(0.5,0.3,"L_{em}=eJ^{#mu}_{em}A_{#mu} , J^{#mu}_{em}=#bar{I}\
   #gamma_{#mu}I , M^{j}_{i}=#SigmaA_{#alpha}#tau^{#alphaj}_{i}");
   c1->Print("latex2.ps");
```

graphics/latex3.C




```
   TCanvas *c1 = new TCanvas("c1");
   TPaveText *pt = new TPaveText(.05,.1,.95,.8);

   pt->AddText("#frac{2s}{#pi#alpha^{2}}  #frac{d#sigma}{dcos#theta} \
   (e^{+}e^{-} #rightarrow f#bar{f} ) = #left| #frac{1}{1 - #Delta#alpha} \
   #right|^{2} (1+cos^{2}#theta)");

   pt->AddText("+ 4 Re #left{ #frac{2}{1 - #Delta#alpha} #chi(s) \
   #[]{#hat{g}_{#nu}^{e}#hat{g}_{#nu}^{f} \
(1 + cos^{2}#theta) + 2 #hat{g}_{a}^{e}#hat{g}_{a}^{f} cos#theta) } #right}");

   pt->AddText("+ 16#left|#chi(s)#right|^{2}\
 #left[(#hat{g}_{a}^{e}^{2} + #hat{g}_{v}^{e}^{2})\
 (#hat{g}_{a}^{f}^{2} + #hat{g}_{v}^{f}^{2})(1+cos^{2}#theta)\
 + 8 #hat{g}_{a}^{e} #hat{g}_{a}^{f} #hat{g}_{v}^{e} \
 #hat{g}_{v}^{f}cos#theta#right] ");

   pt->SetLabel("Born equation");
```

graphics/latex4.C

```
TLatex l;
l.SetTextSize(0.03);

// Draw the columns titles
l.SetTextAlign(22);
l.DrawLatex(0.165, 0.95, "Lower case");
l.DrawLatex(0.495, 0.95, "Upper case");
l.DrawLatex(0.825, 0.95, "Variations");

// Draw the lower case letters
l.SetTextAlign(12);
float y, x1, x2;
y = 0.90; x1 = 0.07; x2 = x1+0.2;
             l.DrawLatex(x1, y, "alpha : ")   ; l.DrawLatex(x2, y, "#alpha");
y -= 0.0375 ; l.DrawLatex(x1, y, "beta : ")    ; l.DrawLatex(x2, y, "#beta");
y -= 0.0375 ; l.DrawLatex(x1, y, "gamma : ")   ; l.DrawLatex(x2, y, "#gamma");
y -= 0.0375 ; l.DrawLatex(x1, y, "delta : ")   ; l.DrawLatex(x2, y, "#delta");
y -= 0.0375 ; l.DrawLatex(x1, y, "epsilon : ") ; l.DrawLatex(x2, y, "#epsilon");
y -= 0.0375 ; l.DrawLatex(x1, y, "zeta : ")    ; l.DrawLatex(x2, y, "#zeta");
y -= 0.0375 ; l.DrawLatex(x1, y, "eta : ")     ; l.DrawLatex(x2, y, "#eta");
y -= 0.0375 ; l.DrawLatex(x1, y, "theta : ")   ; l.DrawLatex(x2, y, "#theta");
y -= 0.0375 ; l.DrawLatex(x1, y, "iota : ")    ; l.DrawLatex(x2, y, "#iota");
```

Lower case		Upper case		Variations	
alpha :	α	Alpha :	A		
beta :	β	Beta :	B		
gamma :	γ	Gamma :	Γ		
delta :	δ	Delta :	Δ		
epsilon :	ϵ	Epsilon :	E	varepsilon :	ε
zeta :	ζ	Zeta :	Z		
eta :	η	Eta :	H		
theta :	θ	Theta :	Θ	vartheta :	ϑ
iota :	ι	Iota :	I		
kappa :	κ	Kappa :	K		
lambda :	λ	Lambda :	Λ		
mu :	μ	Mu :	M		
nu :	ν	Nu :	N		
xi :	ξ	Xi :	Ξ		
omicron :	ο	Omicron :	O		
pi :	π	Pi :	Π		
rho :	ρ	Rho :	P		
sigma :	σ	Sigma :	Σ	varsigma :	ς
tau :	τ	Tau :	T		
upsilon :	υ	Upsilon :	Υ	varUpsilon :	ϒ
phi :	ϕ	Phi :	Φ	varphi :	φ
chi :	χ	Chi :	X		
psi :	ψ	Psi :	Ψ		
omega :	ω	Omega :	Ω	varomega :	ϖ

graphics/latex5.C

```
TLatex l;
l.SetTextSize(0.03);

// Draw First Column
l.SetTextAlign(12);
float y, step, x1, x2;
y = 0.96; step = 0.0465; x1 = 0.02; x2 = x1+0.04;
            l.DrawLatex(x1, y, "#club")           ; l.DrawL
y -= step ; l.DrawLatex(x1, y, "#voidn")          ; l.DrawL
y -= step ; l.DrawLatex(x1, y, "#leq")            ; l.DrawL
y -= step ; l.DrawLatex(x1, y, "#approx")         ; l.DrawL
y -= step ; l.DrawLatex(x1, y, "#in")             ; l.DrawL
y -= step ; l.DrawLatex(x1, y, "#supset")         ; l.DrawL
y -= step ; l.DrawLatex(x1, y, "#cap")            ; l.DrawL
y -= step ; l.DrawLatex(x1, y, "#ocopyright")     ; l.DrawL
```

♣ #club	♦ #diamond	♥ #heart	♠ #spade
℘ #voidn	ℵ #aleph	ℑ #Jgothic	ℜ #Rgothic
≤ #leq	≥ #geq	⟨ #LT	⟩ #GT
≈ #approx	≠ #neq	≡ #equiv	∝ #propto
∈ #in	∉ #notin	⊂ #subset	⊄ #notsubset
⊃ #supset	⊆ #subseteq	⊇ #supseteq	∅ #oslash
∩ #cap	∪ #cup	∧ #wedge	∨ #vee
© #ocopyright	© #copyright	® #oright	® #void1
™ #trademark	™ #void3	Å #AA	å #aa
× #times	÷ #divide	± #pm	⁄ #/
• #bullet	° #circ	… #3dots	· #upoint
ƒ #voidb	∞ #infty	∇ #nabla	∂ #partial
″ #doublequote	∠ #angle	↵ #downleftarrow	¬ #corner
| #lbar	| #cbar	¯ #topbar	⌇ #ltbar
⌊ #arcbottom	⌈ #arctop	⌈ #arcbar	⌊ #bottombar
↓ #downarrow	← #leftarrow	↑ #uparrow	→ #rightarrow
↔ #leftrightarrow	⊗ #otimes	⊕ #oplus	√ #surd
⇓ #Downarrow	⇐ #Leftarrow	⇑ #Uparrow	⇒ #Rightarrow
⇔ #Leftrightarrow	∏ #prod	∑ #sum	∫ #int
| #void8	□ #Box	⊥ #perp	⊙ #odot
ℏ #hbar	‖ #parallel		

graphics/tmathtext.C

```
   TMathText l;
   l.SetTextAlign(23);
   l.SetTextSize(0.06);
   l.DrawMathText(0.50, 1.000, "\\prod_{j\\ge0} \\left(\\sum_{k\\ge0} a_{jk}z^k\
\\right) = \\sum_{n\\ge0} z^n \\left(\\sum_{k_0,k_1,\\ldots\\ge0\\atop k_0+k_1+\\
\cdots=n} a_{0k_0}a_{1k_1} \\cdots \\right)");
```

$$\prod_{j\ge0}\left(\sum_{k\ge0} a_{jk}z^k\right) = \sum_{n\ge0} z^n \left(\sum_{\substack{k_0,k_1,\ldots\ge0\\ k_0+k_1+\cdots=n}} a_{0k_0}a_{1k_1}\cdots\right)$$

$$W^{3\beta}_{\delta_1\rho_1\sigma_2} = U^{3\beta}_{\delta_1\rho_1\sigma_2} + \frac{1}{8\pi^2}\int_{\alpha_1}^{\alpha_2} d\alpha_2' \left[\frac{U^{2\beta}_{\delta_1\rho_1} - \alpha_2' U^{1\beta}_{\rho_1\sigma_2}}{U^{0\beta}_{\rho_1\sigma_2}}\right]$$

$$d\Gamma = \frac{1}{2m_A}\left(\prod_f \frac{d^3p_f}{(2\pi)^3}\frac{1}{2E_f}\right)|\mathscr{M}(m_A - \{p_f\})|^2(2\pi)^4\delta^{(4)}(p_A - \sum p_f)$$

$$4\mathrm{Re}\left\{\frac{2}{1-\Delta\alpha}\chi(s)\left[\hat{g}^e_v\hat{g}^f_v(1+\cos^2\theta) + \hat{g}^e_a\hat{g}^f_a\cos\theta\right]\right\}$$

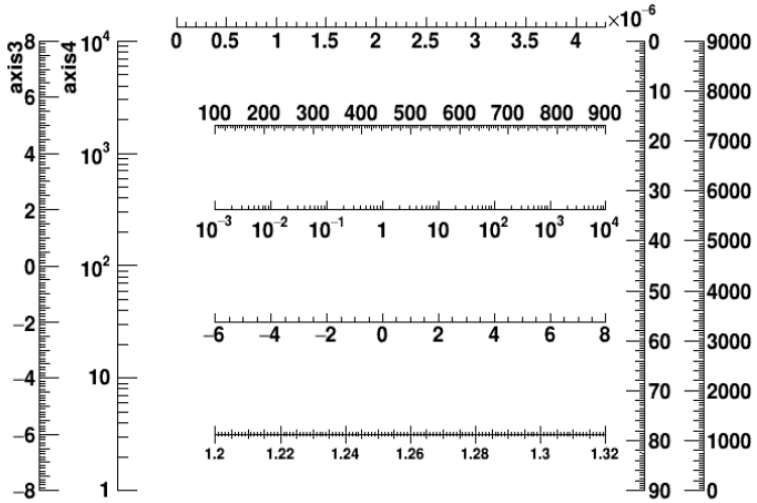
$$p(n) = \frac{1}{\pi\sqrt{2}}\sum_{k=1}^{\infty}\sqrt{k}A_k(n)\frac{d}{dn}\frac{\sinh\left\{\frac{\pi}{k}\sqrt{\frac{2}{3}}\sqrt{n-\frac{1}{24}}\right\}}{\sqrt{n-\frac{1}{24}}}$$

$\frac{(\ell+1)C_\ell^{TE}}{2\pi}$ $\mathbb{N}\subset\mathbb{R}$ RHIC スピン物理 Нью-Йорк

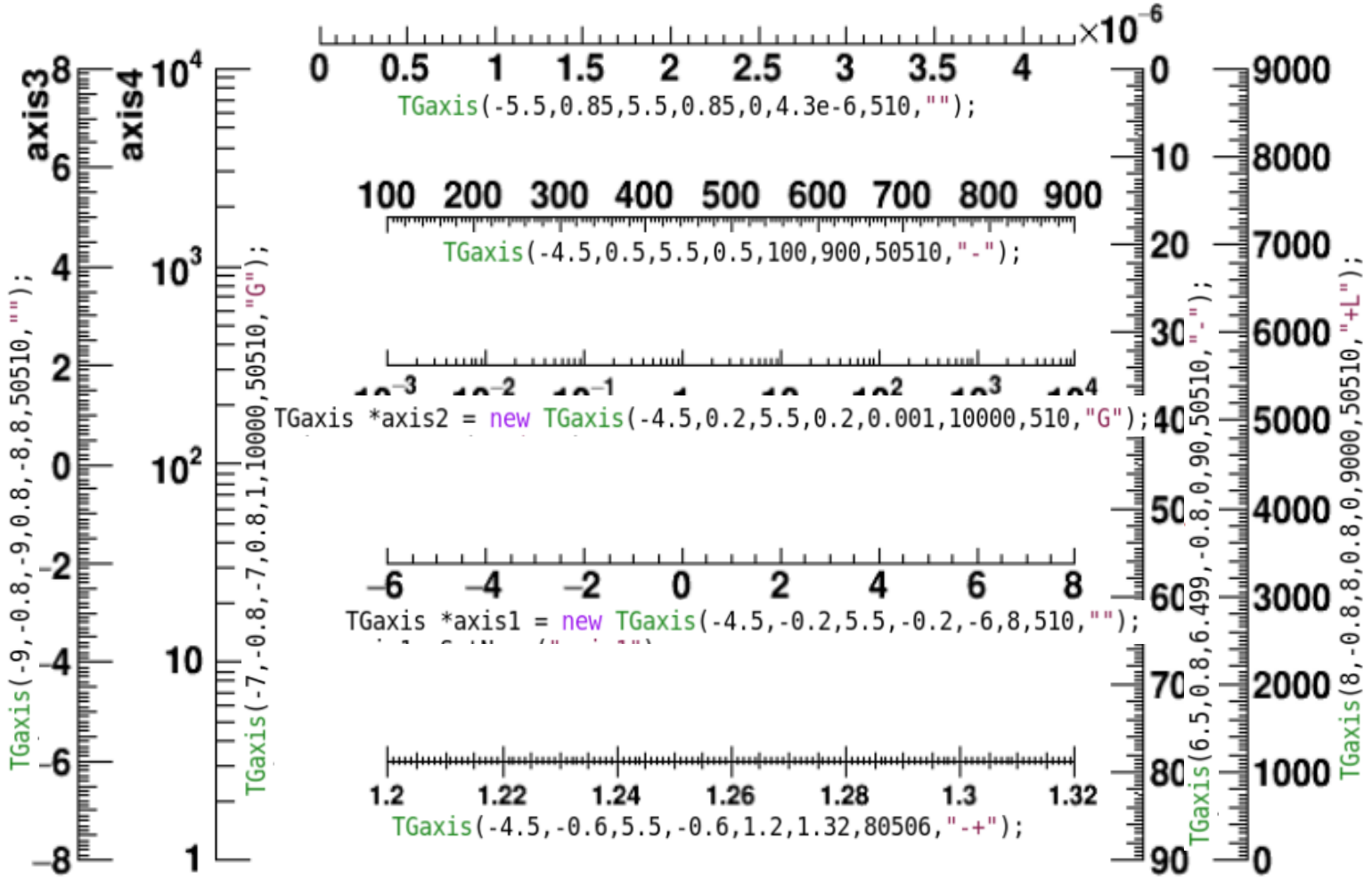
graphics/tmathtext2.C

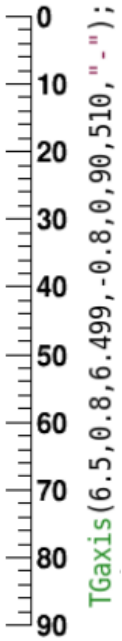
```
TMathText l;
l.SetTextSize(0.013);
l.SetTextFont(132);

// Draw 1st Column
l.SetTextAlign(12);
float y, step, x1, x2;
y = 0.99; step = 0.0126; x1 = 0.01; x2 = x1+0.02;
                l.DrawMathText(x1, y, "\\\""); l.DrawText(x2, y, "\\\"");
y -= step ; l.DrawMathText(x1, y, "\\'"); l.DrawText(x2, y, "\\'");
y -= step ; l.DrawMathText(x1, y, "\\."); l.DrawText(x2, y, "\\.");
y -= step ; l.DrawMathText(x1, y, "\\="); l.DrawText(x2, y, "\\=");
y -= step ; l.DrawMathText(x1, y, "\\Box"); l.DrawText(x2, y, "\\Box");
y -= step ; l.DrawMathText(x1, y, "\\Bumpeq"); l.DrawText(x2, y, "\\Bumpeq");
y -= step ; l.DrawMathText(x1, y, "\\Cap"); l.DrawText(x2, y, "\\Cap");
y -= step ; l.DrawMathText(x1, y, "\\Cup"); l.DrawText(x2, y, "\\Cup");
```

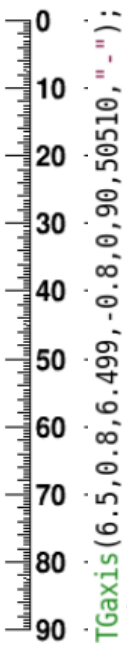
graphics/gaxis.C

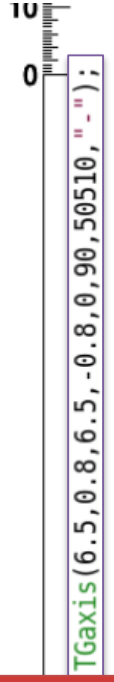



```
c1 = new TCanvas("c1","Examples of Gaxis",10,10,700,500);

c1->Range(-10,-1,10,1);

TGaxis *axis1 = new TGaxis(-4.5,-0.2,5.5,-0.2,-6,8,510,"");
axis1->SetName("axis1");
axis1->Draw();

TGaxis *axis2 = new TGaxis(-4.5,0.2,5.5,0.2,0.001,10000,510,"G");
axis2->SetName("axis2");
axis2->Draw();
```


TGaxis(-5.5,0.85,5.5,0.85,0,4.3e-6,510,"");
TGaxis(-4.5,0.5,5.5,0.5,100,900,50510,"-");
TGaxis *axis2 = new TGaxis(-4.5,0.2,5.5,0.2,0.001,10000,510,"G");
TGaxis *axis1 = new TGaxis(-4.5,-0.2,5.5,-0.2,-6,8,510,"");
TGaxis(-4.5,-0.6,5.5,-0.6,1.2,1.32,80506,"-+");
TGaxis(-9,-0.8,-9,0.8,-8,8,50510,"");
TGaxis(-7,-0.8,-7,0.8,1,10000,50510,"G");
TGaxis(6.5,0.8,6.499,-0.8,0,90,50510,"-");
TGaxis(8,-0.8,8,0.8,0,9000,50510,"+L");
axis3
axis4







Root's problem

void PaintAxis(Double_t xmin, Double_t ymin, Double_t xmax, Double_t ymax, Double_t& wmin, Double_t& wmax, Int_t& ndiv, Option_t* chopt = "", Double_t gridlength = 0, Bool_t drawGridOnly = kFALSE)

Control function to draw an axis.

xmin : X origin coordinate in WC space.
xmax : X end axis coordinate in WC space.
ymin : Y origin coordinate in WC space.
ymax : Y end axis coordinate in WC space.
wmin : Lowest value for the tick mark labels written on the axis.
wmax : Highest value for the tick mark labels written on the axis.
ndiv : Number of divisions.

ndiv=N1 + 100*N2 + 10000*N3
N1=number of 1st divisions.
N2=number of 2nd divisions.
N3=number of 3rd divisions.
e.g.:
nndi=0 --> no tick marks.
nndi=2 --> 2 divisions, one tick mark in the middle of the axis.

chopt : options (see below).

chopt='G': loGarithmic scale, default is linear.
chopt='B': Blank axis. Useful to superpose axis.

Orientation of tick marks on axis.
Tick marks are normally drawn on the positive side of the axis,
however, if x0=x1, then negative.

chopt='+': tick marks are drawn on Positive side. (default)
chopt='-': tick mark are drawn on the negative side.
i.e: '+-' --> tick marks are drawn on both sides of the axis.
chopt='U': Unlabeled axis, default is labeled.

Size of tick marks

By default, tick marks have a length equal to 3 per cent of the axis length.
When the option "S" is specified, the length of the tick marks is equal to fTickSize*axis_length, where fTickSize may be set via TGaxis::SetTickSize.

Position of labels on axis.

Labels are normally drawn on side opposite to tick marks.
However:
chopt='=': on Equal side

Orientation of labels on axis.

Labels are normally drawn parallel to the axis.
However if x0=x1, then Orthogonal
if y0=Y1, then Parallel

Position of labels on tick marks.
Labels are centered on tick marks.
However , if x0=x1, then they are right adjusted.

chopt='R': labels are Right adjusted on tick mark.
(default is centered)
chopt='L': labels are Left adjusted on tick mark.
chopt='C': labels are Centered on tick mark.
chopt='M': In the Middle of the divisions.

Format of labels.

Blank characters are stripped, and then the label is correctly aligned. the dot, if last character of the string, is also stripped, unless the option "." (a dot, or period) is specified.
if SetDecimals(kTRUE) has been called (bit TAxis::kDecimals set). all labels have the same number of decimals after the "." The same is true if gStyle->SetStripDecimals(kFALSE) has been called.

In the following, we have some parameters, like tick marks length and characters height (in percentage of the length of the axis (WC)). The default values are as follows:
Primary tick marks: 3.0 %
Secondary tick marks: 1.5 %
Third order tick marks: .75 %
Characters height for labels: 4%

Labels offset: 1.0 %

optional grid.
chopt='W': cross-Wire
In case of a log axis, the grid is only drawn for the primary tick marks if the number of secondary and tertiary divisions is 0.

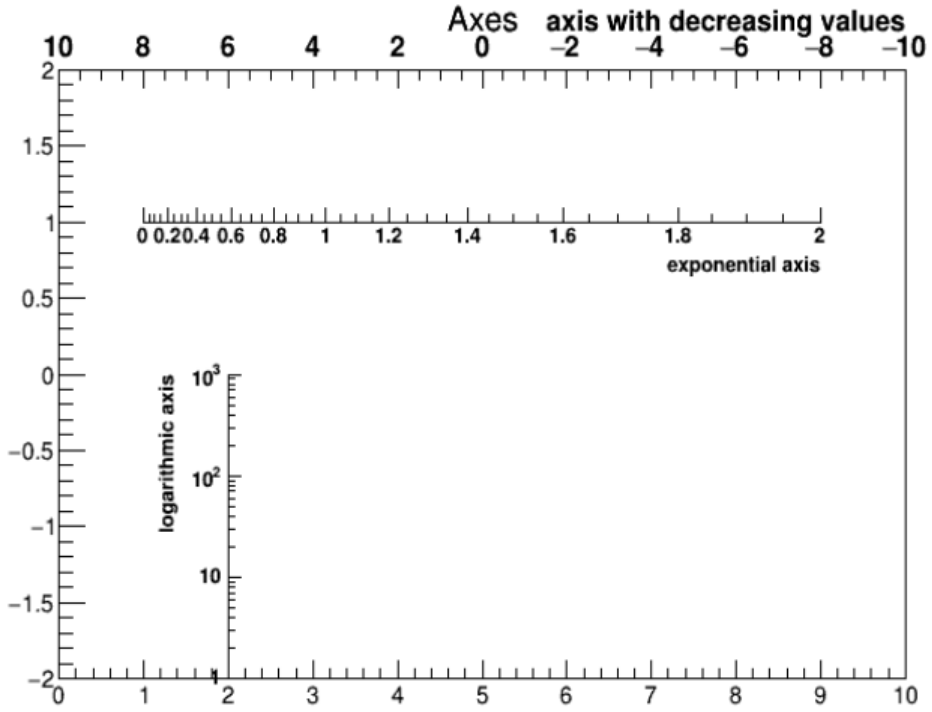
Axis bining optimization.
By default the axis bining is optimized .
chopt='N': No bining optimization
chopt='I': Integer labelling

Maximum Number of Digits for the axis labels See the static function TGaxis::SetMaxDigits

Time representation.
Axis labels may be considered as times, plotted in a defined time format. The format is set with SetTimeFormat(). wmin and wmax are considered as two time values in seconds.
The time axis will be spread around the time offset value (set with SetTimeOffset()). Actually it will go from TimeOffset+wmin to TimeOffset+wmax.
see examples in tutorials timeonaxis.C and timeonaxis2.C

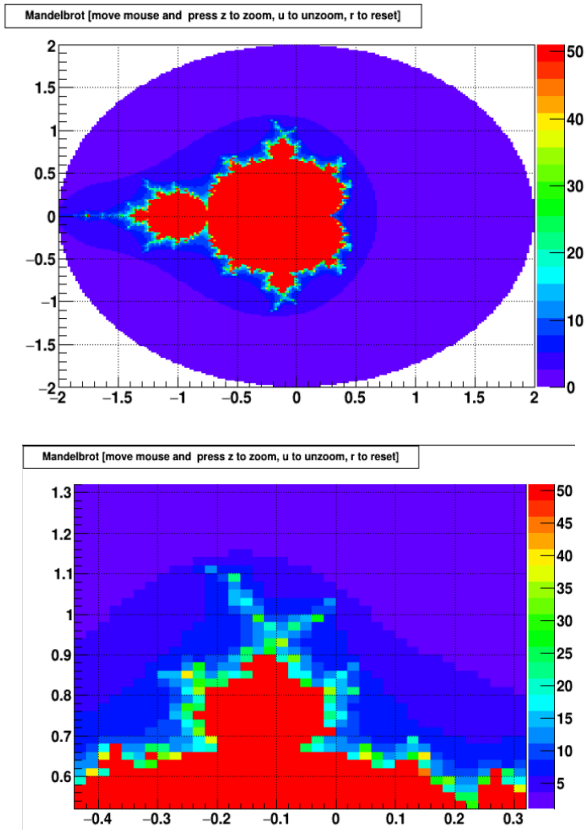
chopt='t': Plot times with a defined format instead of values

/tutorials/graphics/gaxis2.C




```
gStyle->SetOptStat(0);

TH2F *h2 = new TH2F("h","Axes",100,0,10,100,-2,2);
h2->Draw();

TF1 *f1=new TF1("f1","-x",-10,10);
TGaxis *A1 = new TGaxis(0,2,10,2,"f1",510,"-");
A1->SetTitle("axis with decreasing values");
A1->Draw();

TF1 *f2=new TF1("f2","exp(x)",0,2);
TGaxis *A2 = new TGaxis(1,1,9,1,"f2");
A2->SetTitle("exponential axis");
A2->SetLabelSize(0.03);
A2->SetTitleSize(0.03);
A2->SetTitleOffset(1.2);
A2->Draw();

TF1 *f3=new TF1("f3","log10(x)",1,1000);
TGaxis *A3 = new TGaxis(2,-2,2,0,"f3",505,"G");
A3->SetTitle("logarithmic axis");
A3->SetLabelSize(0.03);
A3->SetTitleSize(0.03);
A3->SetTitleOffset(1.2);
A3->Draw();
```

graphics/mandelbrot.C




```
//=============================================================
//
// Using TExec to handle keyboard events and TComplex to draw the Mandelbrot set.
// Author : Luigi Bardelli [ bardelli@fi.infn.it ]
//
// Pressing the keys 'z' and 'u' will zoom and unzoom the picture
// near the mouse location, 'r' will reset to the default view.
//
// Try it (in compiled mode!) with:   root mandelbrot.C+
//
// Details:
//    when a mouse event occurs the myexec() function is called (by
//    using AddExec). Depending on the pressed key, the mygenerate()
//    function is called, with the proper arguments. Note the
//    last_x and last_y variables that are used in myexec() to store
//    the last pointer coordinates (px is not a pointer position in
//    kKeyPress events).
//=============================================================
```

```
TH2F *last_histo=NULL;

void mygenerate(double factor, double cen_x, double cen_y)
{
  printf("Regenerating...\n");
  // resize histo:
  if(factor>0)
    {
      double dx=last_histo->GetXaxis()->GetXmax()-last_histo->GetXaxis()->GetXmin();
      double dy=last_histo->GetYaxis()->GetXmax()-last_histo->GetYaxis()->GetXmin();
      last_histo->SetBins(
                          last_histo->GetNbinsX(),
                          cen_x-factor*dx/2,
                          cen_x+factor*dx/2,
                          last_histo->GetNbinsY(),
                          cen_y-factor*dy/2,
                          cen_y+factor*dy/2
                          );
      last_histo->Reset();
    }
  else
```

If zoom, reset the range Then re-fill

```
    {
      if(last_histo!=NULL) delete last_histo;
      // allocate first view...
      last_histo= new TH2F("h2",
         "Mandelbrot [move mouse and  press z to zoom, u to unzoom, r to reset]",
                           200,-2,2,200,-2,2);
      last_histo->SetStats(0);
    }
  const int max_iter=50;
  for(int bx=1;bx<=last_histo->GetNbinsX();bx++)
    for(int by=1;by<=last_histo->GetNbinsY();by++)
      {
        double x=last_histo->GetXaxis()->GetBinCenter(bx);
        double y=last_histo->GetYaxis()->GetBinCenter(by);
        TComplex point( x,y);
        TComplex z=point;
        int iter=0;
        while (z.Rho()<2){
          z=z*z+point;
          last_histo->Fill(x,y);
          iter++;
          if(iter>max_iter) break;
        }
      }
  last_histo->Draw("colz");
  gPad->Modified();
  gPad->Update();
  printf("Done.\n");
}
```

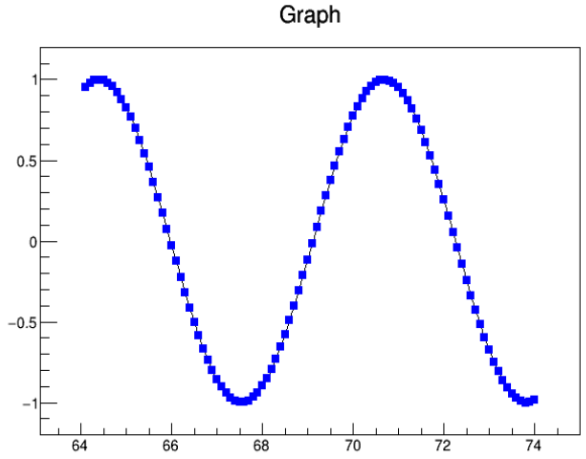
re-fill

```
void myexec()
{
  // get event information
  int event = gPad->GetEvent();
  int px = gPad->GetEventX();
  int py = gPad->GetEventY();

  // some magic to get the coordinates...
  double xd = gPad->AbsPixeltoX(px);
  double yd = gPad->AbsPixeltoY(py);
  float x = gPad->PadtoX(xd);
  float y = gPad->PadtoY(yd);

  static float last_x;
  static float last_y;
```

```
if(event!=kKeyPress)
  {
    last_x=x;
    last_y=y;
    return;
  }

const double Z=2.;
switch(px){
case 'z': // ZOOM
  mygenerate(1./Z, last_x, last_y);
  break;
case 'u': // UNZOOM
  mygenerate(Z   , last_x, last_y);
  break;
case 'r': // RESET
  mygenerate(-1  , last_x, last_y);
  break;
};
```

```
void mandelbrot()
{
  // cosmetics...
  gROOT->SetStyle("Plain");
  gStyle->SetPalette(1,0);
  gStyle->SetPadGridX(kTRUE);
  gStyle->SetPadGridY(kTRUE);
  new TCanvas("canvas","View Mandelbrot set");
  gPad->SetCrosshair();
  // this generates and draws the first view...
  mygenerate(-1,0,0);

  // add exec
  gPad->AddExec("myexec","myexec()");
}
```

graphics/gtime.C



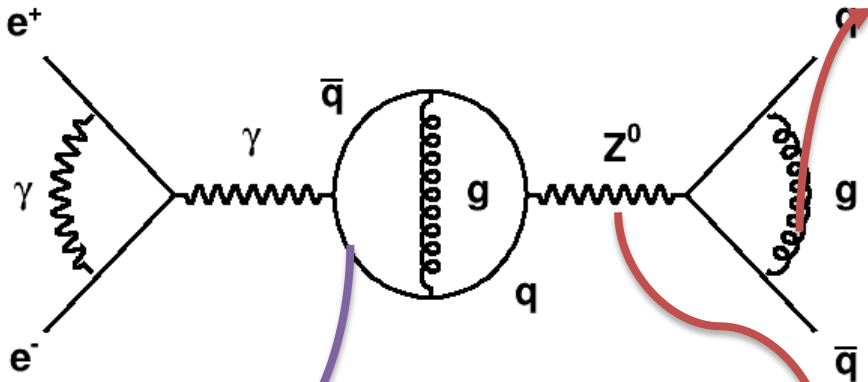

```
// Example of a graph of data moving in time
// Use the canvas "File/Quit" to exit from this example
//Author: Olivier Couet
void gtime() {
   TCanvas *c1 = new TCanvas("c1");
   const Int_t ng = 100;
   const Int_t kNMAX = 2000;
   Double_t *X = new Double_t[kNMAX];
   Double_t *Y = new Double_t[kNMAX];
   Int_t cursor = kNMAX;
   TGraph *g = new TGraph(ng);
   g->SetMarkerStyle(21);
   g->SetMarkerColor(kBlue);
   Double_t x = 0;
   .....
     g->DrawGraph(ng,&X[cursor],&Y[cursor],"alp");
   }
   c1->Update();
   gSystem->ProcessEvents();
   gSystem->Sleep(10);
```

Process pending events (GUI, timers, sockets). Returns the result of TROOT::IsInterrupted(). The interrupt flag (TROOT::SetInterrupt()) can be set during the handling of the events. This mechanism allows macros running in tight calculating loops to be interrupted by some

GUI event (depending on the interval with which this method is called). For example hitting ctrl-c in a canvas will set the interrupt flag.

Sleep(UInt_t milliSec)

graphics/feynman.C




```
TCurlyArc *gluon1 = new TCurlyArc(110, 30, 12.5*sqrt(2), 315, 45);
gluon1->Draw();
```

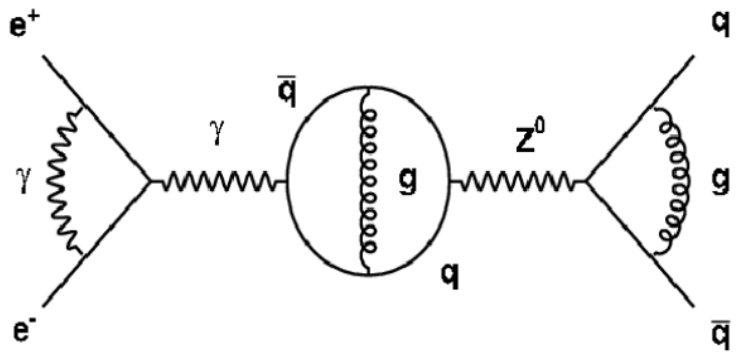
TCurlyArc(Double_t x1, Double_t y1, Double_t rad, Double_t phimin, Double_t phimax, Double_t wl = .02, Double_t amp = .01)
create a new TCurlyarc with center (x1, y1) and radius rad.
The wavelength and amplitude are given in percent of the line length phimin and phimax are given in degrees.

```
TCurlyLine *z0 = new TCurlyLine(85, 30, 110, 30);
z0->SetWavy();
z0->Draw();
t.DrawLatex(100, 37.5,"Z^{0}");
```

```
TArc *a = new TArc(70, 30, 15);
a->Draw();
t.DrawLatex(55, 45,"#bar{q}");
t.DrawLatex(85, 15,"q");
TCurlyLine *gluon = new TCurlyLine(70, 45, 70, 15);
gluon->Draw();
```

graphics/psview.C

The picture below has been loaded from a PS file:

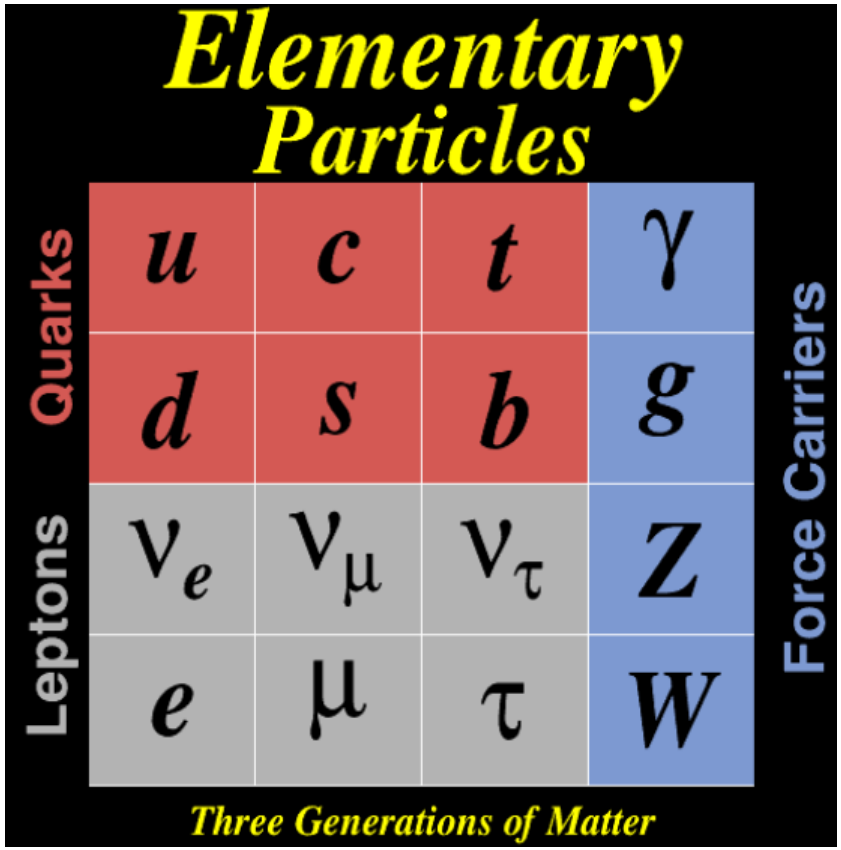



```
// An example how to display PS, EPS, PDF files in canvas
// To load a PS file in a TCanvas, the ghostscript program needs to be install.
// - On most unix systems it is installed by default.
// - On Windows it has to be installed from
http://pages.cs.wisc.edu/~ghost/
//   also the place where gswin32c.exe sits should be added in the PATH.
//One  way to do it is:
//     1. Start the Control Panel
//     2. Double click on System
//     3, Open the "Advanced" tab
//     4. Click on the "Environment Variables" button
//     5. Find "Path" in "System varibale list", click on it.
//     6. Click on the "Edit" button.
//     7. In the "Variable value" field add the path of gswin32c
//        (after a ";") it should be something like:
//        "C:\Program Files\gs\gs8.13\bin"
//     8. click "OK" as much as needed.
```

```
// set to batch mode -> do not display graphics
gROOT->SetBatch(1);

// create a PostScript file
gROOT->Macro("feynman.C");
gPad->Print("feynman.eps");

// back to graphics mode
gROOT->SetBatch(0);

// create an image from PS file
TImage *ps = TImage::Open("feynman.eps");

if (!ps) {
   printf("GhostScript (gs) program must be installed\n");
   return;
}

new TCanvas("psexam", "Example how to display PS file in canvas", 600, 400);
TLatex *tex = new TLatex(0.06,0.9,"The picture below has been loaded from a PS file:");
tex->Draw();
TPad *eps = new TPad("eps", "eps", 0., 0., 1., 0.75);
eps->Draw();
eps->cd();
ps->Draw("xxx");
```

No need option

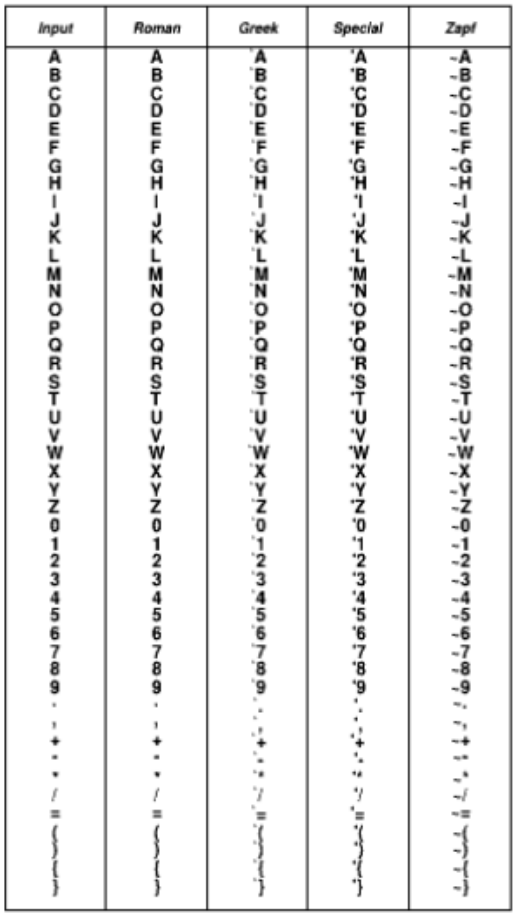
graphics/quarks.C

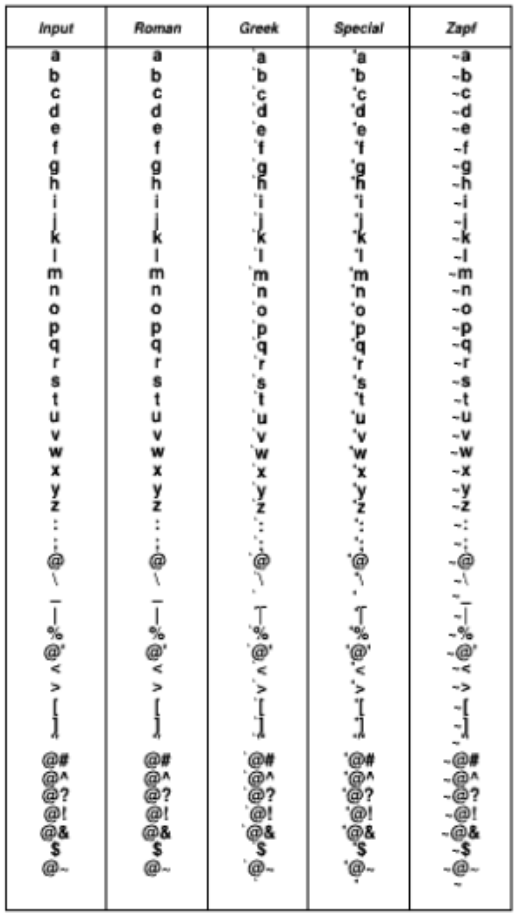



```
TCanvas *c1 = new TCanvas("c1", "c1",10,10,630,760);
c1->SetFillColor(kBlack);
Int_t quarkColor  = 50;
Int_t leptonColor = 16;
Int_t forceColor  = 38;
Int_t titleColor  = kYellow;
Int_t border = 8;

TLatex *texf = new TLatex(0.90,0.455,"Force Carriers");
texf->SetTextColor(forceColor);
texf->SetTextAlign(22); texf->SetTextSize(0.07);
texf->SetTextAngle(90);
texf->Draw();

TLatex *texl = new TLatex(0.11,0.288,"Leptons");
texl->SetTextColor(leptonColor);
texl->SetTextAlign(22); texl->SetTextSize(0.07);
texl->SetTextAngle(90);
texl->Draw();
```

```
// ------------>Create main pad and its subdivisions
   TPad *pad = new TPad("pad", "pad",0.15,0.11,0.85,0.79);
   pad->Draw();
   pad->cd();
   pad->Divide(4,4,0.0003,0.0003);

   pad->cd(1); gPad->SetFillColor(quarkColor);
   gPad->SetBorderSize(border);
   tex.DrawLatex(.5,.5,"u");

   pad->cd(2); gPad->SetFillColor(quarkColor);
   gPad->SetBorderSize(border);
   tex.DrawLatex(.5,.5,"c");

   pad->cd(3); gPad->SetFillColor(quarkColor);
   gPad->SetBorderSize(border);
   tex.DrawLatex(.5,.5,"t");
```

graphics/pstable.C





More plots!

siguang@pku.edu.cn

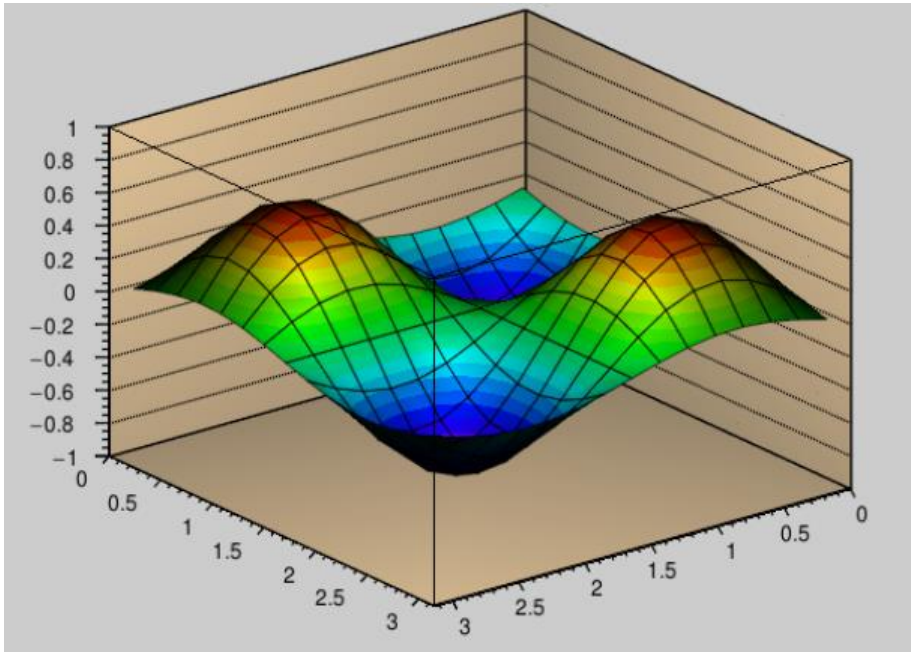
```
// Display all possible types of ROOT/Postscript characters

   char *symbol1[] =
      {"A","B","C","D","E","F","G","H","I","J","K","L","M","N",
       "O","P","Q","R","S","T","U","V","W","X","Y","Z",
       "0","1","2","3","4","5","6","7","8","9",
       ".",",",";","+","-","*","/","=","(",")","{","}","END"};

   char *symbol2[] =
      {"a","b","c","d","e","f","g","h","i","j","k","l","m","n",
       "o","p","q","r","s","t","u","v","w","x","y","z",
       ":","\;","\@","\\","\_","\|","\%",
       "@'","<",">","[","]","\42","@\43","@\136",
       "@\77","@\41","@&","$","@\176"," ","END"};

   char *symbol3[] =
      {"\241","\242","\243","\244","\245","\246","\247","\250",
       "\251","\252","\253","\254","\255","\256","\257","\260",
       "\261","\262","\263","\264","\265","\266","\267","\270",
```

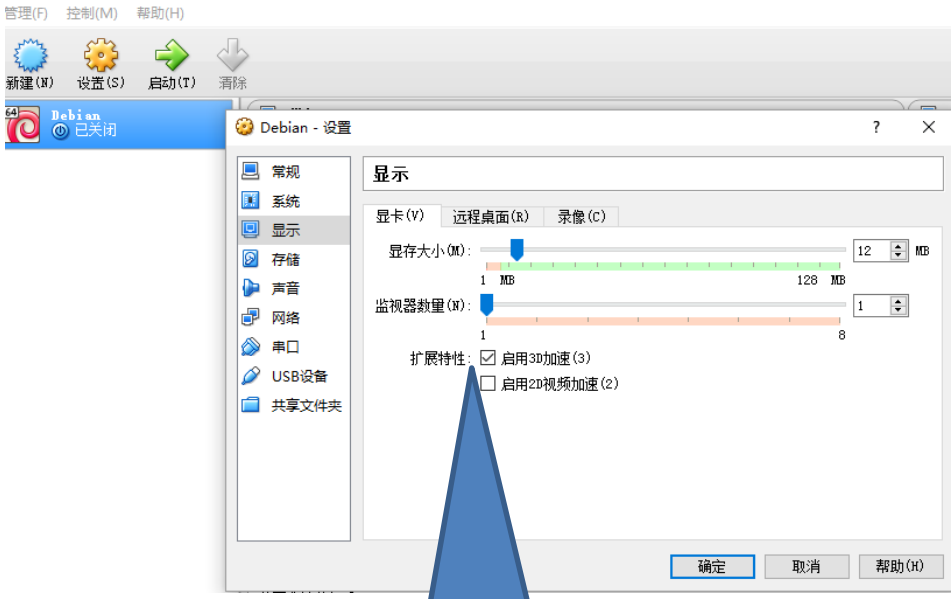
```
TCanvas *c1 = new TCanvas("c1","c1",200,10,w,h);
c1->Range(0,0,xrange,yrange);

TText *t = new TText(0,0,"a");
t->SetTextSize(0.02);
t->SetTextFont(62);
t->SetTextAlign(22);

table(0.5,0.5*xrange-0.5,yrange,t,symbol1,0);
table(0.5*xrange+0.5,xrange-0.5,yrange,t,symbol2,0);
TText *tlabel = new TText(0,0,"a");
tlabel->SetTextFont(72);
tlabel->SetTextSize(0.018);
tlabel->SetTextAlign(22);
tlabel->DrawText(0.5*xrange,1.3,
   "Input characters are standard keyboard characters");
c1->Modified();
c1->Update();
c1->Print("pstable1.ps");
```

```
void table(Float_t x1, Float_t x2, Float_t yrange, TText *t,
   char **symbol, Bool_t octal)
{
   Int_t i;
   Int_t n = 0;
   for (i=0;i<1000;i++) {
      if (!strcmp(symbol[i],"END")) break;
      n++;
   }
```

```
TText *tit = new TText(0,0,"a");
tit->SetTextSize(0.015);
tit->SetTextFont(72);
tit->SetTextAlign(22);
tit->DrawText(xc0,y2-0.6,"Input");
tit->DrawText(xc1,y2-0.6,"Roman");
tit->DrawText(xc2,y2-0.6,"Greek");
tit->DrawText(xc3,y2-0.6,"Special");
tit->DrawText(xc4,y2-0.6,"Zapf");
char text[12];
for (i=0;i<n;i++) {
   if (octal) {
      unsigned char value = *symbol[i];
      sprintf(text,"@\\ %3o",value);
   } else {
      strcpy(text,symbol[i]);
   }
   t->DrawText(xc0,y,text);
   sprintf(text,"%s",symbol[i]);
   t->DrawText(xc1,y,text);
   sprintf(text,"`%s",symbol[i]);
   t->DrawText(xc2,y,text);
   sprintf(text,"'%s",symbol[i]);
```

graphics/anim.C






```
root [0]
Processing anim.C...
OpenGL Warning: Failed to connect to host. Make sure 3D acceleration is enabled for this VM.
```

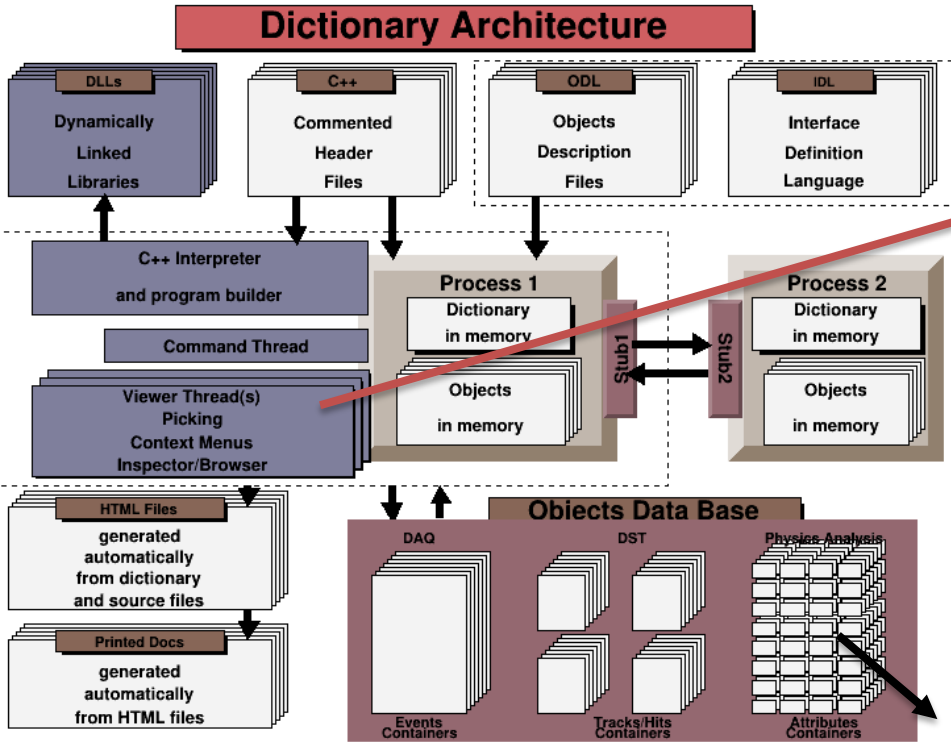
Solved!

```
void anim()
{
   gStyle->SetCanvasPreferGL(true);
   gStyle->SetFrameFillColor(42);
   TCanvas *c1 = new TCanvas("c1");
   c1->SetFillColor(17);
   pi = TMath::Pi();
   f2 = new TF2("f2","sin(2*x)*sin(2*y)*[0]",0,pi,0,pi);
   f2->SetParameter(0,1);
   f2->SetNpx(15);
   f2->SetNpy(15);
   f2->SetMaximum(1);
   f2->SetMinimum(-1);
   f2->Draw("glsurf1");
   TTimer *timer = new TTimer(20);
   timer->SetCommand("Animate()");
   timer->TurnOn();
}
```

```
root [0] TTime T(

TTime TTime()
TTime TTime(Long64_t msec)
TTime TTime(const TTime& t)
```

```
void Animate()
{
   //just in case the canvas has been deleted
   if (!gROOT->GetListOfCanvases()->FindObject("c1")) return;
   t += 0.05*pi;
   f2->SetParameter(0,TMath::Cos(t));
   phi += 2;
   gPad->SetPhi(phi);
   gPad->Modified();
   gPad->Update();
}
```

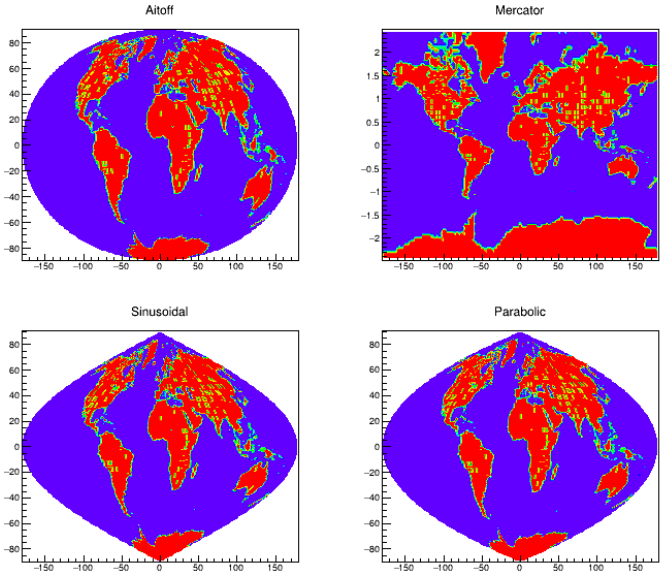
graphics/archi.C




```
TPavesText view(1.0,9.5,7.7,12.6,3,"tr");
view.SetFillColor(39);
view.SetBorderSize(2);
view.SetTextSize(0.023);
view.AddText("Viewer Thread(s)");
view.AddText("Picking");
view.AddText("Context Menus");
view.AddText("Inspector/Browser");
view.Draw();
```

```
for (Int_t j=1;j<9;j++) {
   Float_t y0 = ylow + (j-1)*0.7;
   Float_t y1 = y0 + dy;
   for (Int_t i=1;i<5;i++) {
      Float_t x0 = xlow +(i-1)*0.6;
      Float_t x1 = x0 + dx;
      TPavesText *anal = new TPavesText(x0,y0,x1,y1,7,"tr");
      anal.Draw();
   }
}
```

graphics/earth.C



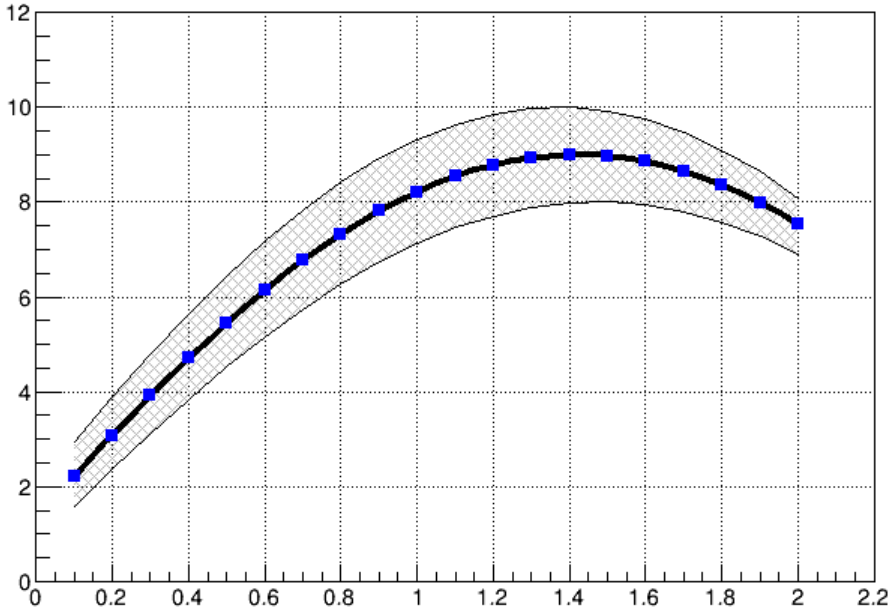

```
//this tutorial illustrate the special contour options
//   "AITOFF"    : Draw a contour via an AITOFF projection
//   "MERCATOR"  : Draw a contour via an Mercator projection
//   "SINUSOIDAL" : Draw a contour via an Sinusoidal projection
//   "PARABOLIC"  : Draw a contour via an Parabolic projection
```

```
TH2F *ha = new TH2F("ha","Aitoff",     180, -180, 180, 179, -89.5, 89.5);
TH2F *hm = new TH2F("hm","Mercator",   180, -180, 180, 161, -80.5, 80.5);
TH2F *hs = new TH2F("hs","Sinusoidal",180, -180, 180, 181, -90.5, 90.5);
TH2F *hp = new TH2F("hp","Parabolic", 180, -180, 180, 181, -90.5, 90.5);
```

。。。。Data Prepare。。。。。。

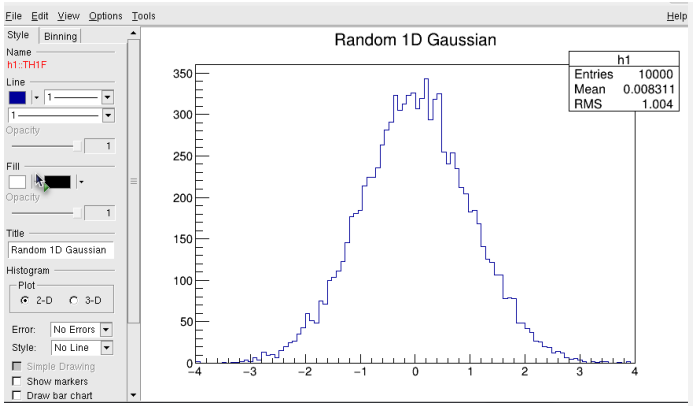
```
c1->cd(1); ha->Draw("aitoff");
c1->cd(2); hm->Draw("mercator");
c1->cd(3); hs->Draw("sinusoidal");
c1->cd(4); hp->Draw("parabolic");
```

graphics/graphShade.C




```
for (i=0;i<n;i++) {
  x[i] = 0.1+i*0.1;
  ymax[i] = 10*sin(x[i]+0.2);
  ymin[i] = 8*sin(x[i]+0.1);
  y[i] = 9*sin(x[i]+0.15);
}
TGraph *grmin = new TGraph(n,x,ymin);
TGraph *grmax = new TGraph(n,x,ymax);
TGraph *gr    = new TGraph(n,x,y);
TGraph *grshade = new TGraph(2*n);
for (i=0;i<n;i++) {
   grshade->SetPoint(i,x[i],ymax[i]);
   grshade->SetPoint(n+i,x[n-i-1],ymin[n-i-1]);
}
grshade->SetFillStyle(3013);
grshade->SetFillColor(16);
grshade->Draw("f");
grmin->Draw("l");
grmax->Draw("l");
gr->SetLineWidth(4);
gr->SetMarkerColor(4);
gr->SetMarkerStyle(21);
gr->Draw("CP");
```

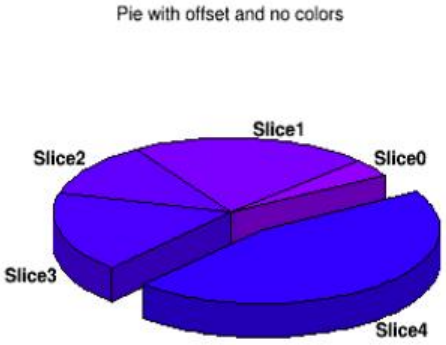
graphics/graph_edit_playback.C

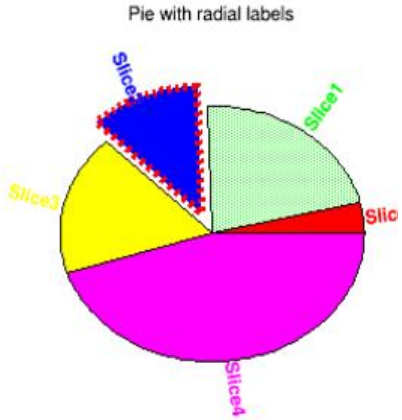


Method to prepare graphedit_playback.root can be checked the comments of the code

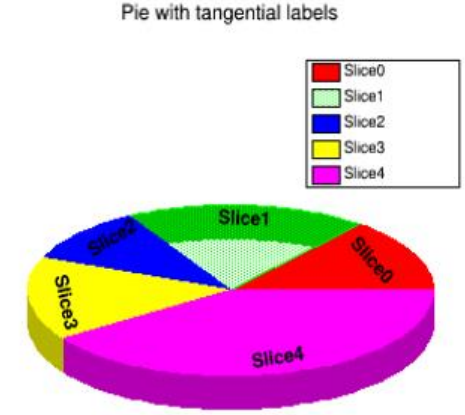
```
Int_t file_size(char *filename)
{
   FileStat_t fs;
   gSystem->GetPathInfo(filename, fs);
   return (Int_t)fs.fSize;
}

void graph_edit_playback()
{
   r = new TRecorder();
   r->Replay("http://root.cern.ch/files/graphedit_playback.root");

   // wait for the recorder to finish the replay
   while (r->GetState() == TRecorder::kReplaying) {
      gSystem->ProcessEvents();
      gSystem->Sleep(1);
   }
```

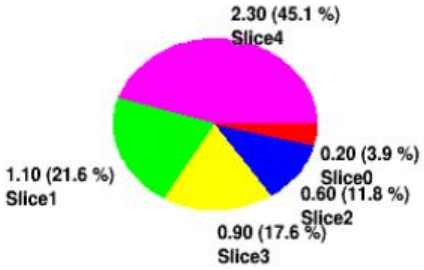
graphics/piechart.C






```
Float_t vals[] = {.2,1.1,.6,.9,2.3};
Int_t colors[] = {2,3,4,5,6};
Int_t nvals = sizeof(vals)/sizeof(vals[0]);

TCanvas *cpie = new TCanvas("cpie","TPie test",700,700);
cpie->Divide(2,2);

TPie *pie1 = new TPie("pie1",
   "Pie with offset and no colors",nvals,vals);
TPie *pie2 = new TPie("pie2",
   "Pie with radial labels",nvals,vals,colors);
TPie *pie3 = new TPie("pie3",
   "Pie with tangential labels",nvals,vals,colors);
TPie *pie4 = new TPie("pie4",
   "Pie with verbose labels",nvals,vals,colors);
```






```
cpie->cd(1);
pie1->SetAngularOffset(30.);
pie1->SetEntryRadiusOffset( 4, 0.1);
pie1->SetRadius(.35);
pie1->Draw("3d");
```

```
cpie->cd(2);
pie2->SetEntryRadiusOffset(2,.05);
pie2->SetEntryLineColor(2,2);
pie2->SetEntryLineWidth(2,5);
pie2->SetEntryLineStyle(2,2);
pie2->SetEntryFillStyle(1,3030);
pie2->SetCircle(.5,.45,.3);
pie2->Draw("rsc");

cpie->cd(3);
pie3->SetY(.32);
pie3->GetSlice(0)->SetValue(.8);
pie3->GetSlice(1)->SetFillStyle(3031);
pie3->SetLabelsOffset(-.1);
pie3->Draw("3d t nol");
TLegend *pieleg = pie3->MakeLegend();
pieleg->SetY1(.56); pieleg->SetY2(.86);
```

SetCircle(Double_t x = .5, Double_t y = .5, Double_t rad = .4)

"R" Print the labels along the central "R"adius of slices.
"T" Print the label in a direction "T"angent to circle that describes the TPie.
"3D" Draw the pie-chart with a pseudo 3D effect.
"NOL" No OutLine: Don't draw the slices' outlines, any property over the slices' line is ignored.
">" Sort the slices in increasing order.
"<" Sort the slices in decreasing order.

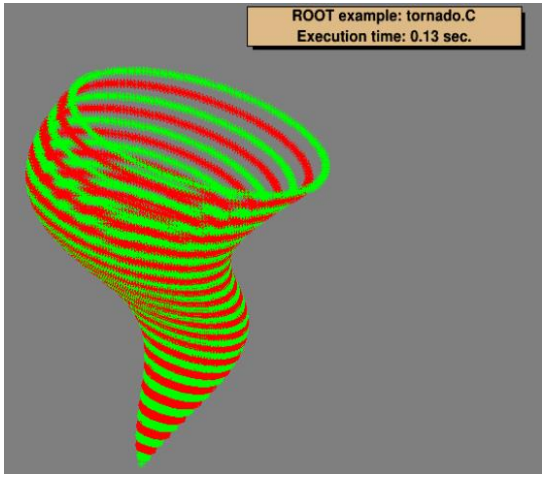
```
cpie->cd(4);
pie4->SetRadius(.2);
pie4->SetLabelsOffset(.01);
pie4->SetLabelFormat("#splitline{%val (%perc)}{%txt}");
pie4->Draw("nol <");
```

- %txt : to print the text label associated with the slice
- %val : to print the numeric value of the slice
- %frac : to print the relative fraction of this slice
- %perc : to print the % of this slice

ex. : mypie->SetLabelFormat("%txt (%frac)");

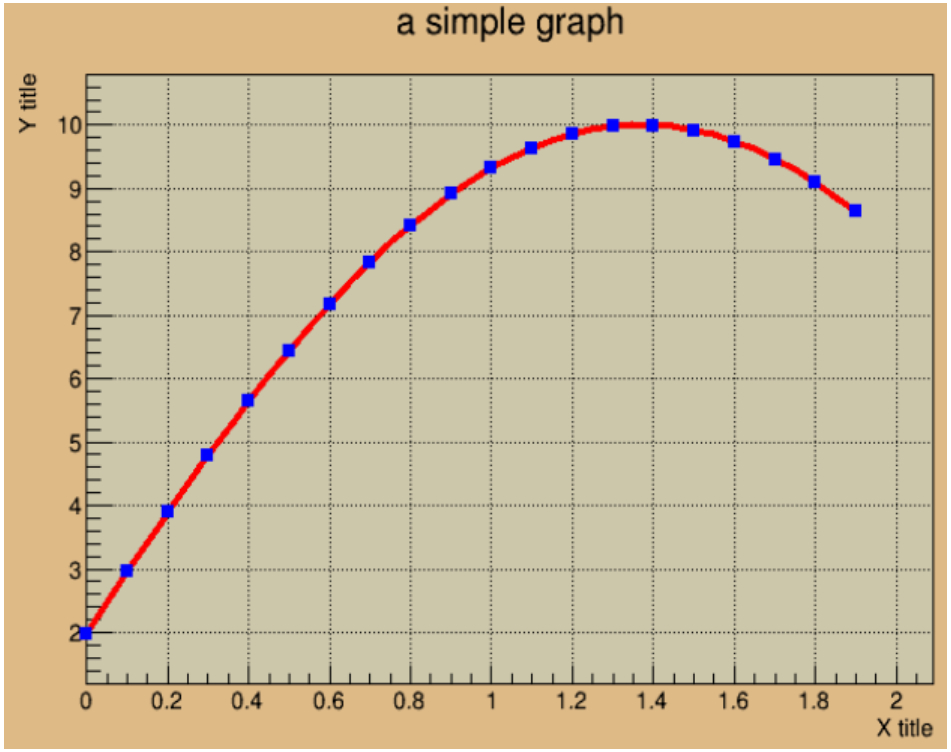
graphics/tornado.C

```
TView *view = TView::CreateView(1,0,0);
float range = numberOfCircles*d;
view->SetRange( 0, 0, 0, 4.0*range, 2.0*range, range );

for( int j = d; j < numberOfCircles*d; j += d ) {

    // create a PolyMarker3D
    TPolyMarker3D *pm3d = new TPolyMarker3D( numberOfPoints );

    float x, y, z;

    // set points
    for( int i = 1; i < numberOfPoints; i++ ) {
        float csin = sin(2*PI / (double)numberOfPoints  * (double)i) + 1;
        float ccos = cos(2*PI / (double)numberOfPoints  * (double)i) + 1;
        float esin = sin(2*PI / (double)(numberOfCircles*d) * (double)j) + 1;
        x = j * ( csin + esin );
        y = j * ccos;
        z = j;
        pm3d->SetPoint( i, x, y, z );
    }

    // set marker size, color & style
    pm3d->SetMarkerSize( 1 );
    pm3d->SetMarkerColor( 2 + ( d == ( j & d ) ) );
    pm3d->SetMarkerStyle( 3 );

    //draw
    pm3d->Draw();
```



Codes under /tutorials/graphs

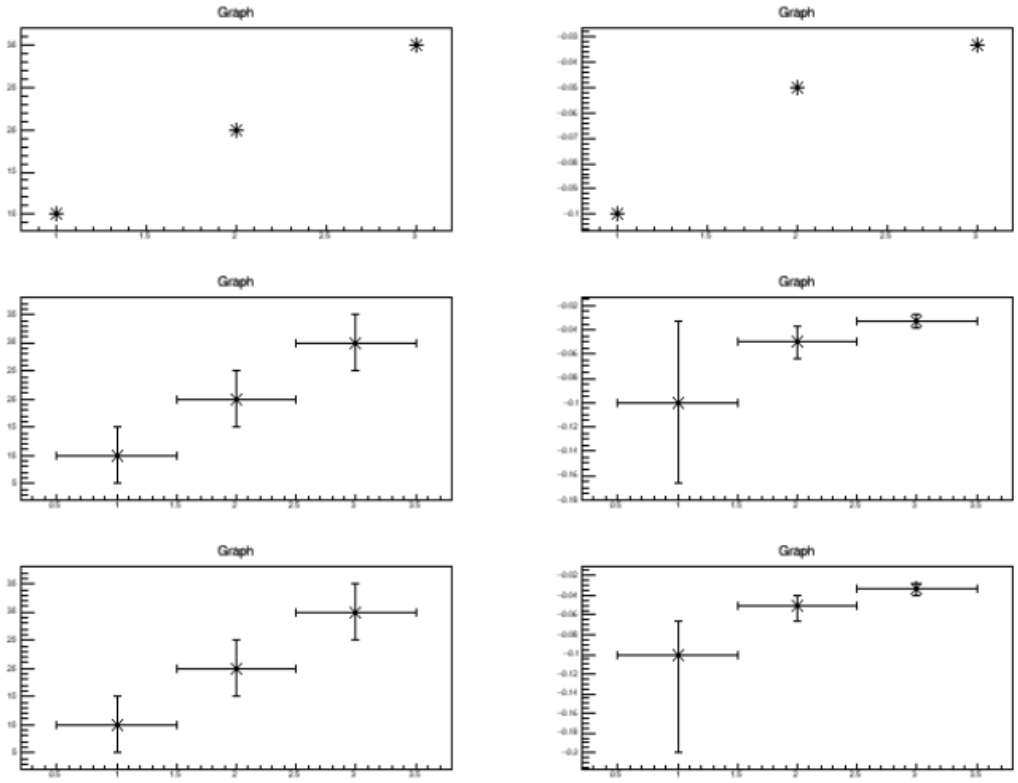
graphs/graph.C




```
const 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);
  printf(" i %i %f %f \n",i,x[i],y[i]);
}
gr = new TGraph(n,x,y);
gr->SetLineColor(2);
gr->SetLineWidth(4);
gr->SetMarkerColor(4);
gr->SetMarkerStyle(21);
gr->SetTitle("a simple graph");
gr->GetXaxis()->SetTitle("X title");
gr->GetYaxis()->SetTitle("Y title");
gr->Draw("ACP");
```

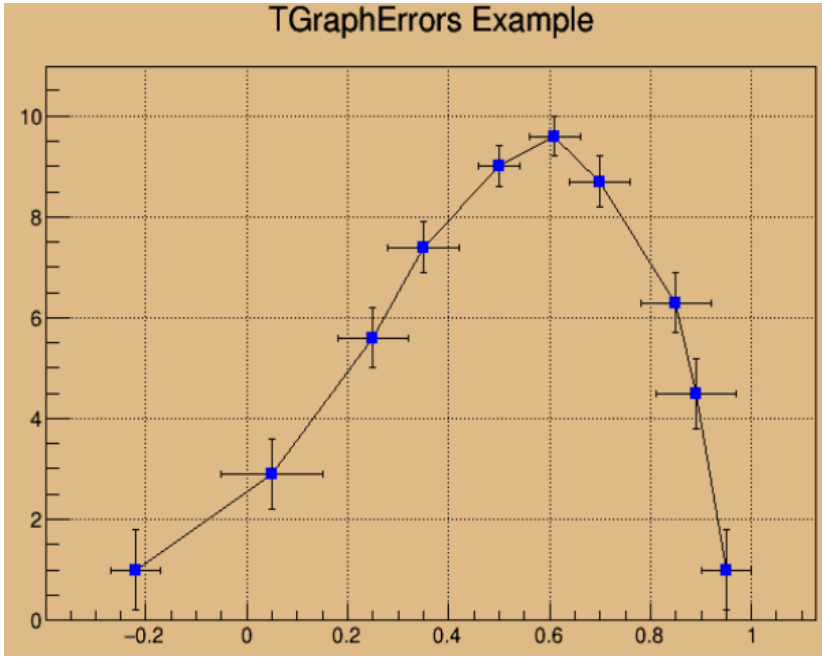
graphs/graphApply.C

```
TGraph *gr1 = new TGraph(npoints,xaxis,yaxis);
TGraphErrors *gr2 = new TGraphErrors(npoints,xaxis,yaxis,errorx,errory);
TGraphAsymmErrors *gr3 = new TGraphAsymmErrors(npoints,xaxis,yaxis,exl,exh,eyl,eyh);
TF2 *ff = new TF2("ff","-1./y");

TCanvas *c1 = new TCanvas("c1","c1");
c1->Divide(2,3);
// TGraph
c1->cd(1);
gr1->DrawClone("A*");
c1->cd(2);
gr1->Apply(ff);
gr1->Draw("A*");

// TGraphErrors
c1->cd(3);
gr2->DrawClone("A*");
c1->cd(4);
gr2->Apply(ff);
gr2->Draw("A*");
```



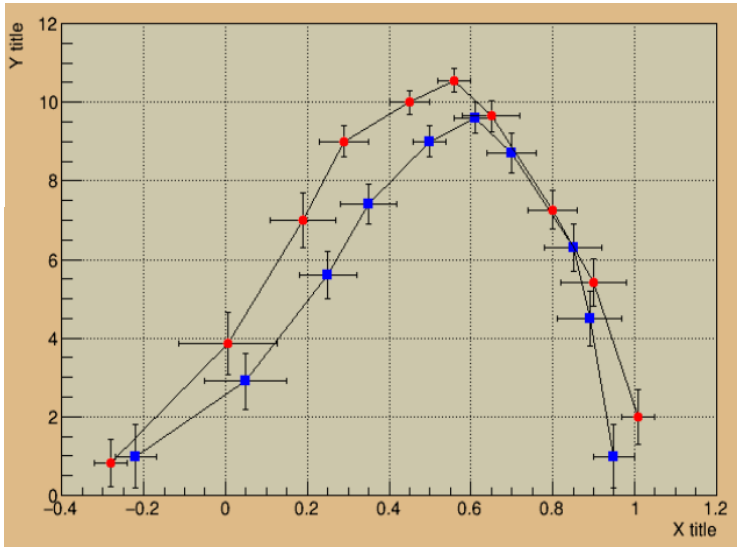
graphs/gerrors.C

```
const Int_t n = 10;
Float_t x[n]  = {-0.22, 0.05, 0.25, 0.35, 0.5, 0.61,0.7,0.85,0.89,0.95};
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");
```



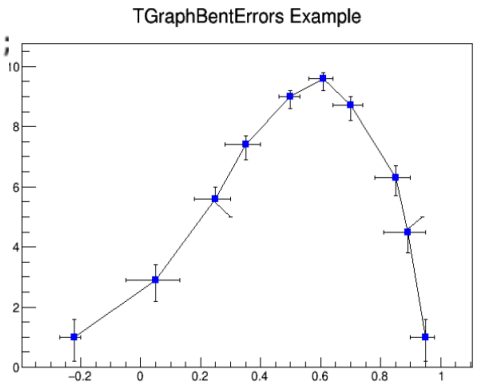
graphs/gerrors2.C

```
TH1F *hr = c1->DrawFrame(-0.4,0,1.2,12);
hr->SetXTitle("X title");
hr->SetYTitle("Y title");
c1->GetFrame()->SetFillColor(21);
c1->GetFrame()->SetBorderSize(12);

   // create first graph
const Int_t n1 = 10;
Double_t x1[]  = {-0.22, 0.05, 0.25, 0.35, 0.5, 0.61,0.7,0.85,0.89,0.95};
Double_t y1[]  = {1,2.9,5.6,7.4,9,9.6,8.7,6.3,4.5,1};
Double_t ex1[] = {.05,.1,.07,.07,.04,.05,.06,.07,.08,.05};
Double_t ey1[] = {.8,.7,.6,.5,.4,.4,.5,.6,.7,.8};
TGraphErrors *gr1 = new TGraphErrors(n1,x1,y1,ex1,ey1);
gr1->SetMarkerColor(kBlue);
gr1->SetMarkerStyle(21);
gr1->Draw("LP");

   // create second graph
const Int_t n2 = 10;
Float_t x2[]  = {-0.28, 0.005, 0.19, 0.29, 0.45, 0.56,0.65
Float_t y2[]  = {0.82,3.86,7,9,10,10.55,9.64,7.26,5.42,2};
Float_t ex2[] = {.04,.12,.08,.06,.05,.04,.07,.06,.08,.04};
Float_t ey2[] = {.6,.8,.7,.4,.3,.3,.4,.5,.6,.7};
TGraphErrors *gr2 = new TGraphErrors(n2,x2,y2,ex2,ey2);
gr2->SetMarkerColor(kRed);
gr2->SetMarkerStyle(20);
gr2->Draw("LP");
```



graphs/bent.C

```
void bent() {
// Bent error bars
   const Int_t n = 10;
   Double_t x[n]  = {-0.22, 0.05, 0.25, 0.35, 0.5, 0.61,0.7,0.85,0.89,0.95};
   Double_t y[n]  = {1,2.9,5.6,7.4,9,9.6,8.7,6.3,4.5,1};
   Double_t exl[n] = {.05,.1,.07,.07,.04,.05,.06,.07,.08,.05};
   Double_t eyl[n] = {.8,.7,.6,.5,.4,.4,.5,.6,.7,.8};
   Double_t exh[n] = {.02,.08,.05,.05,.03,.03,.04,.05,.06,.03};
   Double_t eyh[n] = {.6,.5,.4,.3,.2,.2,.3,.4,.5,.6};
   Double_t exld[n] = {.0,.0,.0,.0,.0,.0,.0,.0,.0,.0};
   Double_t eyld[n] = {.0,.0,.05,.0,.0,.0,.0,.0,.0,.0};
   Double_t exhd[n] = {.0,.0,.0,.0,.0,.0,.0,.0,.0,.0};
   Double_t eyhd[n] = {.0,.0,.0,.0,.0,.0,.0,.0,.05,.0};
   TGraphBentErrors *gr = new TGraphBentErrors(
      n,x,y,exl,exh,eyl,eyh,exld,exhd,eyld,eyhd);
   gr->SetTitle("TGraphBentErrors Example");
   gr->SetMarkerColor(4);
   gr->SetMarkerStyle(21);
   gr->Draw("ALP");
}
```



by default horizonthal and vertical small lines are drawn at the end of the error bars. if option "z" or "Z" is specified, these lines are not drawn.

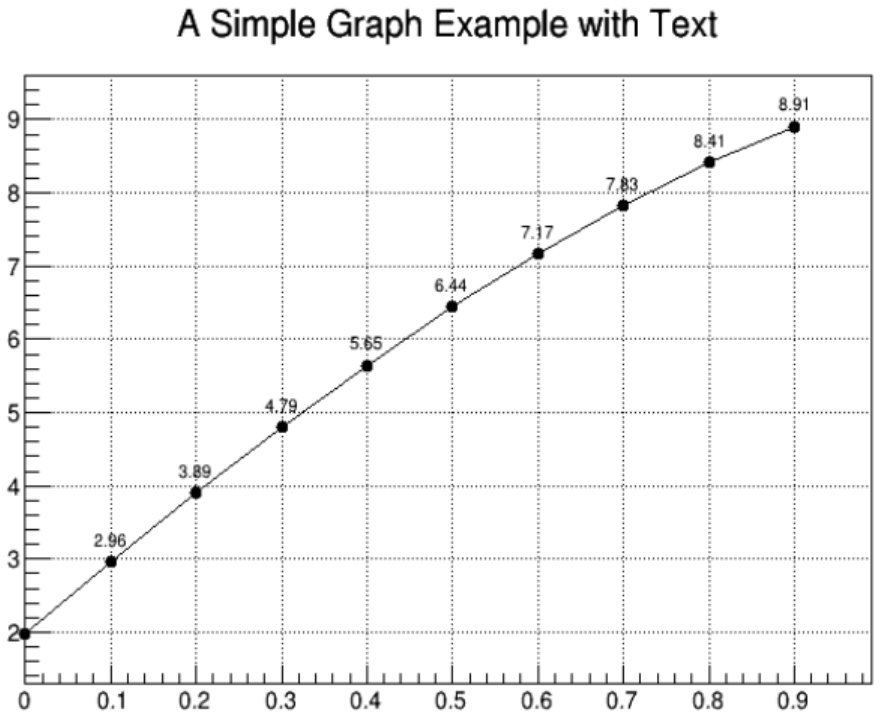
if option contains ">" an arrow is drawn at the end of the error bars
if option contains "|>" a full arrow is drawn at the end of the error bars
the size of the arrow is set to 2/3 of the marker size.

By default, error bars are drawn. If option "X" is specified, the errors are not drawn (TGraph::Paint equivalent).

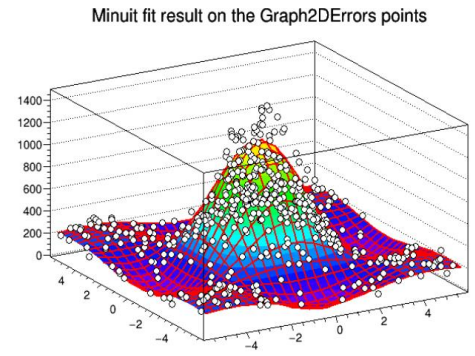
if option "[]" is specified only the end vertical/horizonthal lines of the error bars are drawn. This option is interesting to superimpose systematic errors on top of a graph with statistical errors.

graphs/graphtext.C

```
TGraph *g = (TGraph*)gPad->GetListOfPrimitives()->FindObject("Graph");
n = g->GetN();
for (i=1; i<n; i++) {
   g->GetPoint(i,x,y);
   l = new TLatex(x,y+0.2,Form("%4.2f",y));
   l->SetTextSize(0.025);
   l->SetTextFont(42);
   l->SetTextAlign(21);
   l->Paint();
}
```

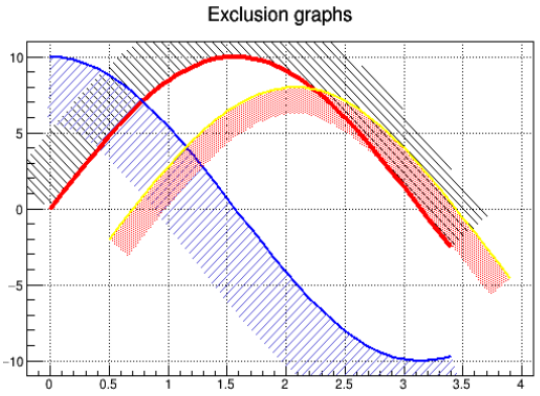


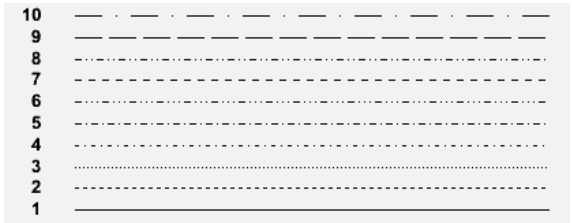
graphs/graph2derrorsfit.C




```
TRandom r;
TF2  *f2 = new TF2("f2","1000*(([0]*sin(x)/x)*([1]*sin(y)/y))+200",-6,6,-6,6);
f2->SetParameters(1,1);
TGraph2DErrors *dte = new TGraph2DErrors(nd);

// Fill the 2D graph
Double_t zmax = 0;
for (Int_t i=0; i<nd; i++) {
   f2->GetRandom2(x,y);
   rnd = r.Uniform(-e,e); // Generate a random number in [-e,e]
   z = f2->Eval(x,y)*(1+rnd);
   if (z>zmax) zmax = z;
   dte->SetPoint(i,x,y,z);
   ex = 0.05*r.Rndm();
   ey = 0.05*r.Rndm();
   ez = TMath::Abs(z*rnd);
   dte->SetPointError(i,ex,ey,ez);
}
f2->SetParameters(0.5,1.5);
dte->Fit(f2);
TF2 *fit2 = (TF2*)dte->FindObject("f2");
fit2->SetTitle("Minuit fit result on the Graph2DErrors points");
fit2->SetMaximum(zmax);
gStyle->SetHistTopMargin(0);
fit2->Draw("surf1");
dte->Draw("same p0");
```

graphs/exclusiongraph.C



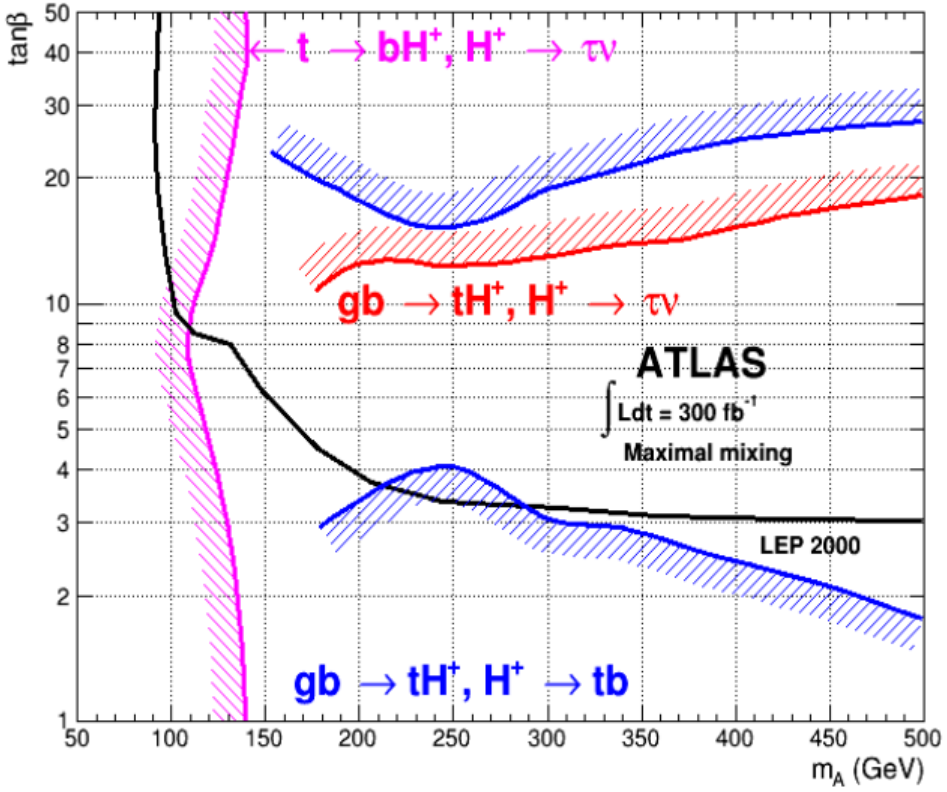



```
TMultiGraph *mg = new TMultiGraph();
mg->SetTitle("Exclusion graphs");
TGraph *gr1 = new TGraph(n,x1,y1);
gr1->SetLineColor(2);
gr1->SetLineWidth(1504);
gr1->SetFillStyle(3005);

TGraph *gr2 = new TGraph(n,x2,y2);
gr2->SetLineColor(4);
gr2->SetLineWidth(-2002);
gr2->SetFillStyle(3004);
gr2->SetFillColor(9);

TGraph *gr3 = new TGraph(n,x3,y3);
gr3->SetLineColor(5);
gr3->SetLineWidth(-802);
gr3->SetFillStyle(3002);
gr3->SetFillColor(2);

mg->Add(gr1);
mg->Add(gr2);
mg->Add(gr3);
mg->Draw("AC");
```

C or L option is needed

This drawing mode is activated when the absolute value of the graph line width (set thanks to SetLineWidth) is greater than 99. In that case the line width number is interpreted as 100*ff+ll = ffll. The two-digit numbers “ll” represent the normal line width whereas “ff” is the filled area width. The sign of “ffll” allows flipping the filled area from one side of the line to the other

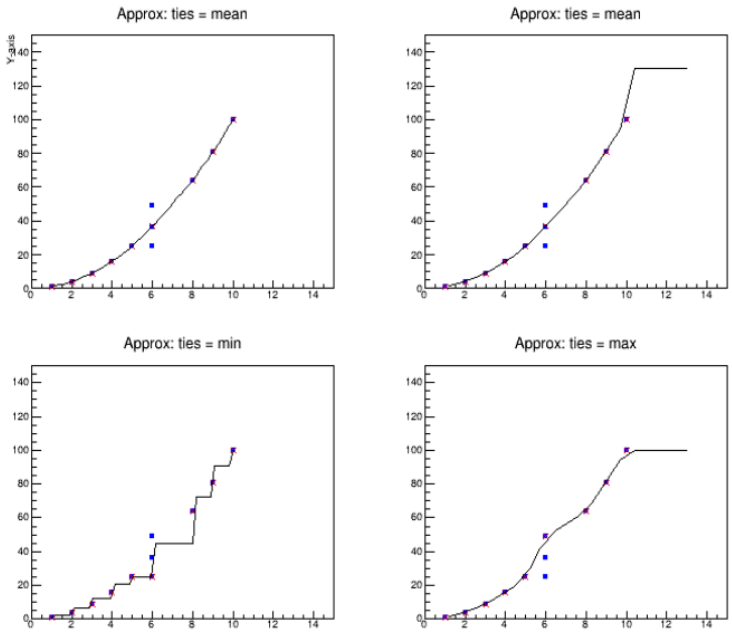
graphs/exclusiongraph2.C




```
TCanvas *c = new TCanvas("c",
   "Charged Higgs L300 Contour",0,0,700,700);
c->SetTickx();
c->SetTicky();
c->SetGridx();
c->SetGridy();

TH1 *frame = new TH1F("frame","",1000,50,500);
frame->SetMinimum(1);
frame->SetMaximum(50);
frame->SetDirectory(0);
frame->SetStats(0);
frame->GetXaxis()->SetTitle("m_{A} (GeV)");
frame->GetXaxis()->SetTickLength(0.02);
frame->GetXaxis()->SetLabelSize(0.03);
frame->GetYaxis()->SetTitle("tan#beta");
frame->GetYaxis()->SetMoreLogLabels();
frame->GetYaxis()->SetLabelSize(0.03);
frame->Draw(" ");
c->SetLogy();
```

```
TGraph *gr1 = new TGraph(10);
gr1->SetFillColor(6);
gr1->SetFillStyle(3005);
gr1->SetLineColor(6);
gr1->SetLineWidth(603);
gr1->SetPoint(0,140,0.5);
gr1->SetPoint(1,130,2.9);
gr1->SetPoint(2,124.677,3.83726);
gr1->SetPoint(3,113.362,6.06903);
gr1->SetPoint(4,108.513,8.00221);
gr1->SetPoint(5,111.746,10.0272);
gr1->SetPoint(6,119.828,12.8419);
gr1->SetPoint(7,135.991,30.0872);
gr1->SetPoint(8,140,40);
gr1->SetPoint(9,135,60);
gr1->Draw("C");
TLatex *tex = new TLatex(140.841,37.9762,
   "#leftarrow t #rightarrow bH^{+}, H^{+} #rightarrow #tau#nu");
tex->SetTextColor(6);
tex->Draw();
```

graphs/approx.C




```
// test data (square)
   Int_t n = 11;
   Double_t x[] = {1,2,3,4,5,6,6,6,8,9,10};
   Double_t y[] = {1,4,9,16,25,25,36,49,64,81,100};
   grxy = new TGraph(n,x,y);

// x values, for which y values should be interpolated
   Int_t nout = 14;
   Double_t xout[] =
      {1.2,1.7,2.5,3.2,4.4,5.2,5.7,6.5,7.6,8.3,9.7,10.4,11.3,13};

// create Canvas
   vC1 = new TCanvas("vC1","square",200,10,700,700);
   vC1->Divide(2,2);
```

```
// initialize graph with data
   grin = new TGraph(n,x,y);
// interpolate at equidistant points (use mean for tied x-values)
   TGraphSmooth *gs = new TGraphSmooth("normal");
   grout = gs->Approx(grin,"linear");
   DrawSmooth(1,"Approx: ties = mean","X-axis","Y-axis");
```

```
// re-initialize graph with data
// (since graph points were set to unique vales)
   grin = new TGraph(n,x,y);
// interpolate at given points xout
   grout = gs->Approx(grin,"linear", 14, xout, 0, 130);
   DrawSmooth(2,"Approx: ties = mean","","");

// print output variables for given values xout
   Int_t vNout = grout->GetN();
   Double_t vXout, vYout;
   for (Int_t k=0;k<vNout;k++) {
      grout->GetPoint(k, vXout, vYout);
      cout << "k= " << k << "  vXout[k]= " << vXout
           << "  vYout[k]= " << vYout << endl;
   }
```

```
// re-initialize graph with data
   grin = new TGraph(n,x,y);
// interpolate at equidistant points (use min for tied x-values)
//   grout = gs->Approx(grin,"linear", 50, 0, 0, 0, 1, 0, "min");
   grout = gs->Approx(grin,"constant", 50, 0, 0, 0, 1, 0.5, "min");
   DrawSmooth(3,"Approx: ties = min","","");

// re-initialize graph with data
   grin = new TGraph(n,x,y);
// interpolate at equidistant points (use max for tied x-values)
   grout = gs->Approx(grin,"linear", 14, xout, 0, 0, 2, 0, "max");
   DrawSmooth(4,"Approx: ties = max","","");

// cleanup
   delete gs;
}
```

TGraph * Approx(TGraph* grin, Option_t* option = "linear", Int_t nout = 50, Double_t* xout = 0, Double_t yleft = 0, Double_t yright = 0, Int_t rule = 0, Double_t f = 0, Option_t* ties = "mean")

grin: graph giving the coordinates of the points to be interpolated. Alternatively a single plotting structure can be specified:

option: specifies the interpolation method to be used. Choices are "linear" (iKind = 1) or "constant" (iKind = 2).

nout: If xout is not specified, interpolation takes place at n equally spaced points spanning the interval [min(x), max(x)], where nout = max(nout, number of input data).

xout: an optional set of values specifying where interpolation is to take place.

yleft: the value to be returned when input x values less than min(x). The default is defined by the value of rule given below.

yright: the value to be returned when input x values greater than max(x). The default is defined by the value of rule given below.

rule: an integer describing how interpolation is to take place outside the interval [min(x), max(x)]. If rule is 0 then the given yleft and yright values are returned, if it is 1 then 0 is returned for such points and if it is 2, the value at the closest data extreme is used.

f: For method="constant" a number between 0 and 1 inclusive, indicating a compromise between left- and right-continuous step functions. If y0 and y1 are the values to the left and right of the point then the value is y0*f+y1*(1-f) so that f=0 is right-continuous and f=1 is left-continuous

ties: Handling of tied x values. An integer describing a function with a single vector argument returning a single number result:

ties = "ordered" (iTies = 0): input x are "ordered"
ties = "mean" (iTies = 1): function "mean"
ties = "min" (iTies = 2): function "min"
ties = "max" (iTies = 3): function "max"

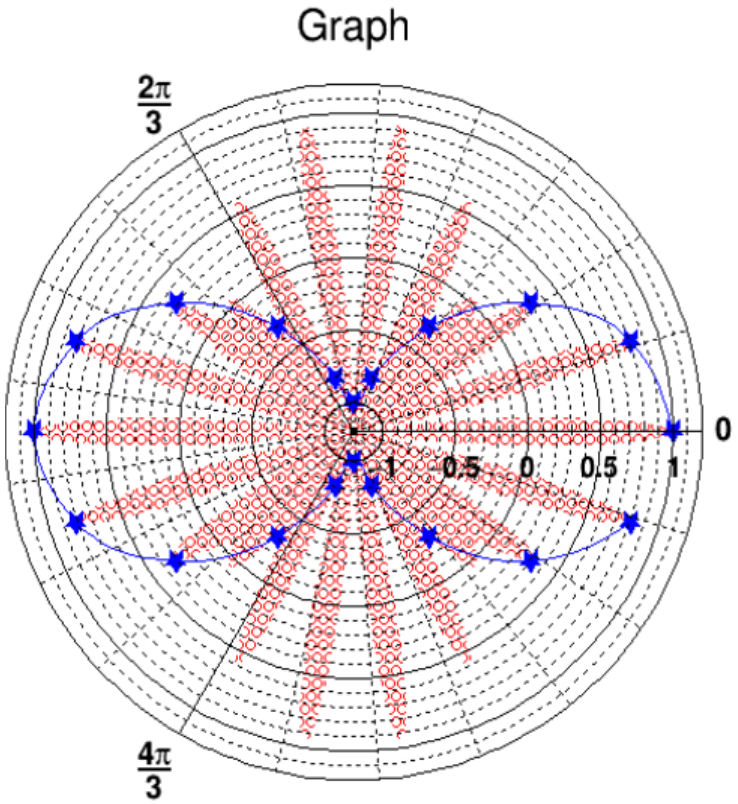
Details:

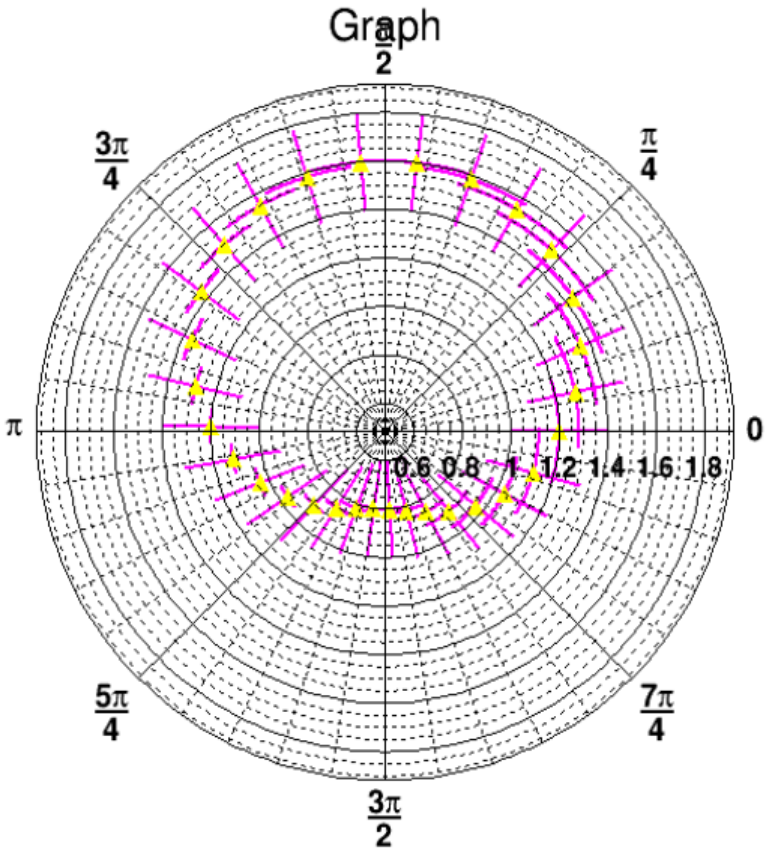
At least two complete (x, y) pairs are required. If there are duplicated (tied) x values and ties is a function it is
applied to the y values for each distinct x value. Useful functions in this context include mean, min, and max.
If ties="ordered" the x values are assumed to be already ordered. The first y value will be used for interpolation to the left and the last one for interpolation to the right.

Value:

approx returns a graph with components x and y, containing n coordinates which interpolate the given data points according to the method (and rule) desired.

graphs/graphpolar.C

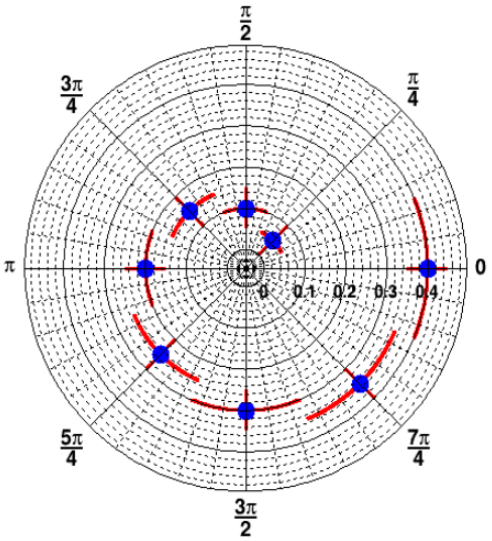




```
TF1 * fplot = new TF1("fplot","cos(2*x)*cos(20*x)",xmin,xmax);

for (Int_t ipt = 0; ipt < 1000; ipt++){
   x[ipt] = ipt*(xmax-xmin)/1000+xmin;
   y[ipt] = fplot->Eval(x[ipt]);
}

TGraphPolar * grP = new TGraphPolar(1000,x,y)
grP->SetLineColor(2);
grP->SetLineWidth(0.2);
grP->SetFillStyle(3012);
grP->SetFillColor(2);
grP->Draw("AFL");
```

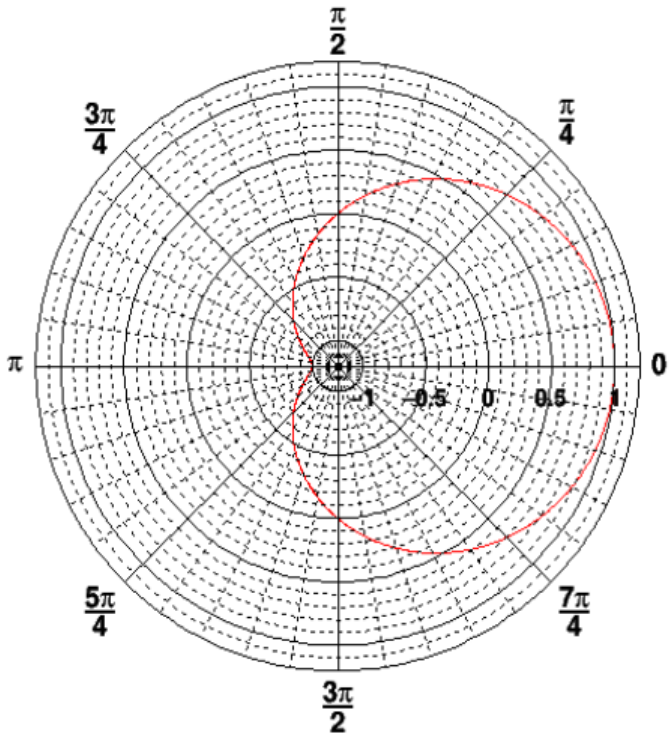
```
Double_t ex[30];
Double_t ey[30];
for (Int_t ipt = 0; ipt < 30; ipt++){
   x2[ipt] = x[1000/30*ipt];
   y2[ipt] = 1.2 + 0.4*sin(TMath::Pi()*2*ipt/30);
   ex[ipt] = 0.2+0.1*cos(2*TMath::Pi()/30*ipt);
   ey[ipt] = 0.2;
}

TGraphPolar * grPE = new TGraphPolar(30,x2,y2,ex,ey);
grPE->SetMarkerStyle(22);
grPE->SetMarkerSize(1.5);
grPE->SetMarkerColor(5);
grPE->SetLineColor(6);
grPE->SetLineWidth(2);
grPE->Draw("EP");
```

graphs/graphpolar2.C




```
for (int i=0; i<8; i++) {
   theta[i]   = (i+1)*(TMath::Pi()/4.);
   radius[i]  = (i+1)*0.05;
   etheta[i]  = TMath::Pi()/8.;
   eradius[i] = 0.05;
}

TGraphPolar * grP1 = new TGraphPolar(8, theta, radius, etheta, eradius);
grP1->SetTitle("");

grP1->SetMarkerStyle(20);
grP1->SetMarkerSize(2.);
grP1->SetMarkerColor(4);
grP1->SetLineColor(2);
grP1->SetLineWidth(3);
grP1->Draw("PE");

CPol->Update();
grP1->GetPolargram()->SetToRadian();
```

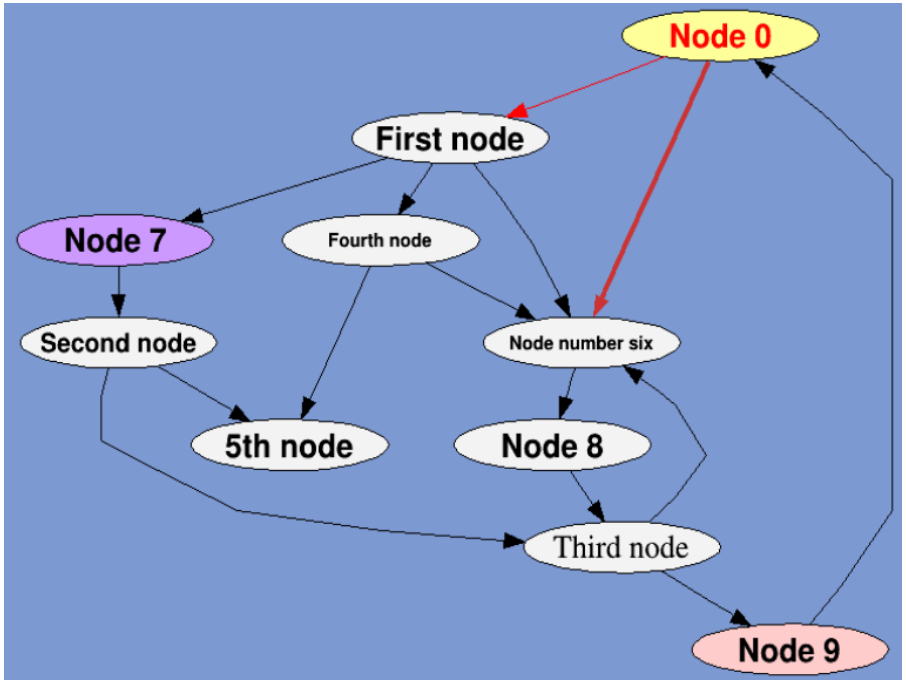
graphs/graphpolar3.C




```
TF1 * fp1 = new TF1("fplot","cos(x)",rmin,rmax);
for (Int_t ipt = 0; ipt < 1000; ipt++) {
   r[ipt] = ipt*(rmax-rmin)/1000+rmin;
   theta[ipt] = fp1->Eval(r[ipt]);
}
TGraphPolar * grP1 = new TGraphPolar(1000,r,theta);
grP1->SetTitle("");
grP1->SetLineColor(2);
grP1->Draw("AOL");
```

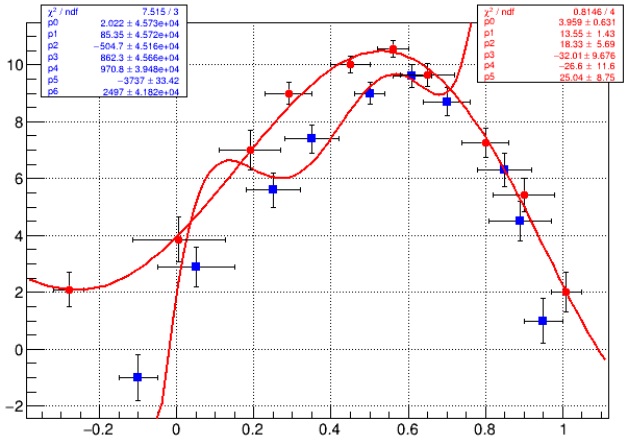
“O” Polar labels are paint orthogonally to the polargram radius.
“P” Polymarker are paint at each point position.
“E” Paint error bars.
“F” Paint fill area (closed polygon).
“A”Force axis redrawing even if a polagram already exists.

graphs/graphstruct.C



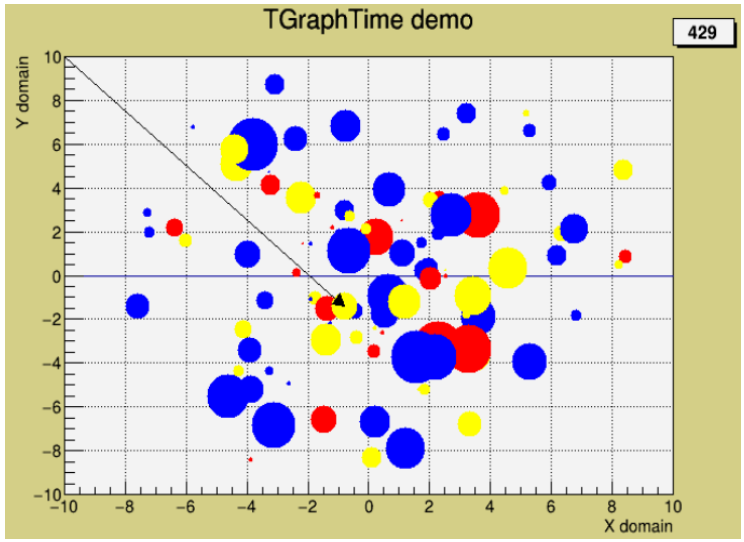

```
TGraphStruct *gs = new TGraphStruct();

// create some nodes and put them in the graph in one go ...
TGraphNode *n0 = gs->AddNode("n0","Node 0");
TGraphNode *n1 = gs->AddNode("n1","First node");
TGraphNode *n2 = gs->AddNode("n2","Second node");
TGraphNode *n3 = gs->AddNode("n3","Third node");
TGraphNode *n4 = gs->AddNode("n4","Fourth node");

  ........
// some edges ...
gs->AddEdge(n0,n1)->SetLineColor(kRed);
TGraphEdge *e06 = gs->AddEdge(n0,n6);
e06->SetLineColor(kRed-3);
e06->SetLineWidth(4);
gs->AddEdge(n1,n7);
gs->AddEdge(n4,n6);
gs->AddEdge(n3,n9);
gs->AddEdge(n6,n8);
gs->AddEdge(n7,n2);
gs->AddEdge(n8,n3);

........
TCanvas *c = new TCanvas("c","c",800,600);
c->SetFillColor(38);
gs->Draw();
```

graphs/multigraph.C



```
   // draw a frame to define the range
TMultiGraph *mg = new TMultiGraph();

TGraphErrors *gr1 = new TGraphErrors(n1,x1,y1,ex1,ey1);
gr1->SetMarkerColor(kBlue);
gr1->SetMarkerStyle(21);
gr1->Fit("pol6","q");
mg->Add(gr1);


TGraphErrors *gr2 = new TGraphErrors(n2,x2,y2,ex2,ey2);
gr2->SetMarkerColor(kRed);
gr2->SetMarkerStyle(20);
gr2->Fit("pol5","q");

mg->Add(gr2);

mg->Draw("ap");

  //force drawing of canvas to generate the fit TPaveStats
c1->Update();
TPaveStats *stats1 = (TPaveStats*)gr1->GetListOfFunctions()->FindObject("stats");
TPaveStats *stats2 = (TPaveStats*)gr2->GetListOfFunctions()->FindObject("stats");
stats1->SetTextColor(kBlue);
stats2->SetTextColor(kRed);
stats1->SetX1NDC(0.12); stats1->SetX2NDC(0.32); stats1->SetY1NDC(0.75);
stats2->SetX1NDC(0.72); stats2->SetX2NDC(0.92); stats2->SetY1NDC(0.78);
```

graphs/gtime.C

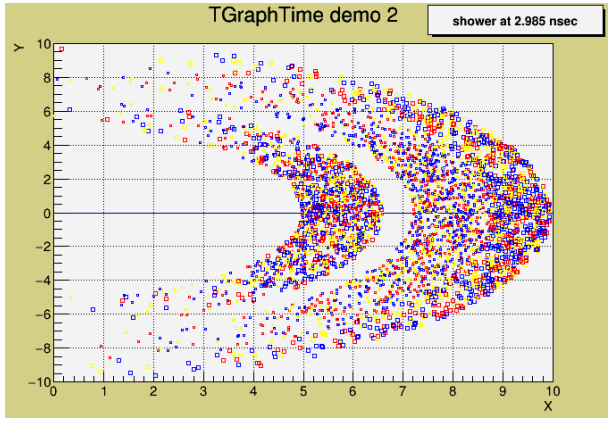



```
void gtime(Int_t nsteps = 500, Int_t np=100) {
   if (np > 1000) np = 1000;
   Int_t color[1000];
   Double_t rr[1000], phi[1000], dr[1000], size[1000];
   TRandom3 r;
   Double_t xmin = -10, xmax = 10, ymin = -10, ymax = 10;
   TGraphTime *g = new TGraphTime(nsteps,xmin,ymin,xmax,ymax);
   g->SetTitle("TGraphTime demo;X domain;Y domain");
   Int_t i,s;
   for (i=0;i<np;i++) { //calculate some object parameters
      rr[i]  = r.Uniform(0.1*xmax,0.2*xmax);
      phi[i] = r.Uniform(0,2*TMath::Pi());
      dr[i]  = r.Uniform(0,1)*0.9*xmax/Double_t(nsteps);
      Double_t rc = r.Rndm();
      color[i] = kRed;
      if (rc > 0.3) color[i] = kBlue;
      if (rc > 0.7) color[i] = kYellow;
      size[i] = r.Uniform(0.5,6);
   }
```

```
for (s=0;s<nsteps;s++) { //fill the TGraphTime step by step
   for (i=0;i<np;i++) {
      Double_t newr = rr[i]+dr[i]*s;
      Double_t newsize = 0.2+size[i]*TMath::Abs(TMath::Sin(newr+10));
      Double_t newphi = phi[i] + 0.01*s;
      Double_t xx = newr*TMath::Cos(newphi);
      Double_t yy = newr*TMath::Sin(newphi);
      TMarker *m = new TMarker(xx,yy,20);
      m->SetMarkerColor(color[i]);
      m->SetMarkerSize(newsize);
      g->Add(m,s);
      if (i==np-1) g->Add(new TArrow(xmin,ymax,xx,yy,0.02,"-|>"), s);
   }
   g->Add(new TPaveLabel(.90,.92,.98,.97,Form("%d",s+1),"brNDC"),s);
}
g->Draw();

//save object to a file
TFile f("gtime.root","recreate");
g->Write("g");
//to view this object in another session do
//  TFile f("gtime.root");
//  g.Draw();
```

Int_t Add(const TObject* obj, Int_t slot, Option_t* option = "")
Add one object to a time slot.

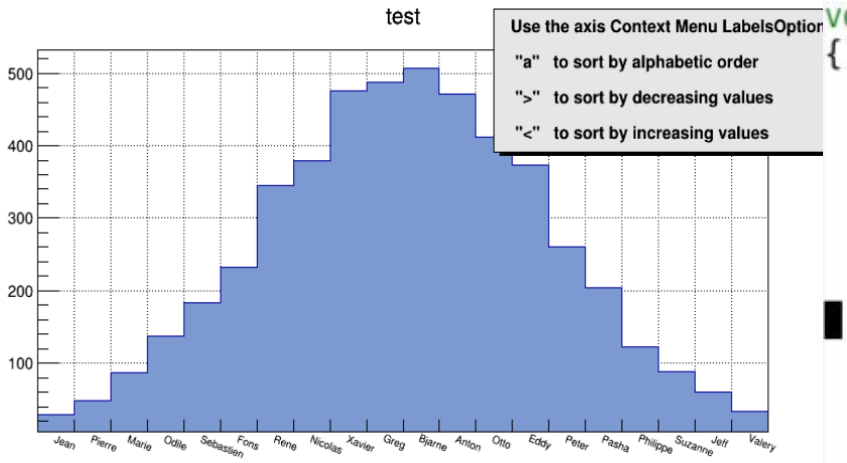
graphs/gtime2.C




```
void gtime2(Int_t nsteps = 200, Int_t np=5000) {
   if (np > 5000) np = 5000;
   Int_t color[5000];
   Double_t cosphi[5000], sinphi[5000], speed[5000];
   TRandom3 r;
   Double_t xmin = 0, xmax = 10, ymin = -10, ymax = 10;
   TGraphTime *g = new TGraphTime(nsteps,xmin,ymin,xmax,ymax);
   g->SetTitle("TGraphTime demo 2;X;Y");
   Int_t i,s;
   Double_t phi,fact = xmax/Double_t(nsteps);
   for (i=0;i<np;i++) { //calculate some object parameters
      speed[i]  = r.Uniform(0.5,1);
      phi       = r.Gaus(0,TMath::Pi()/6.);
      cosphi[i] = fact*speed[i]*TMath::Cos(phi);
      sinphi[i] = fact*speed[i]*TMath::Sin(phi);
      Double_t rc = r.Rndm();
      color[i] = kRed;
      if (rc > 0.3) color[i] = kBlue;
      if (rc > 0.7) color[i] = kYellow;
   }
```

```
   for (s=0;s<nsteps;s++) { //fill the TGraphTime step by step
      for (i=0;i<np;i++) {
         Double_t xx = s*cosphi[i];
         if (xx < xmin) continue;
         Double_t yy = s*sinphi[i];
         TMarker *m = new TMarker(xx,yy,25);
         m->SetMarkerColor(color[i]);
         m->SetMarkerSize(1.5 -s/(speed[i]*nsteps));
         g->Add(m,s);
      }
      g->Add(new TPaveLabel(.70,.92,.98,.99,Form("shower at %5.3f nsec",3.*s/nsteps),"brNDC"),s);
   }
   g->Draw();
}
```

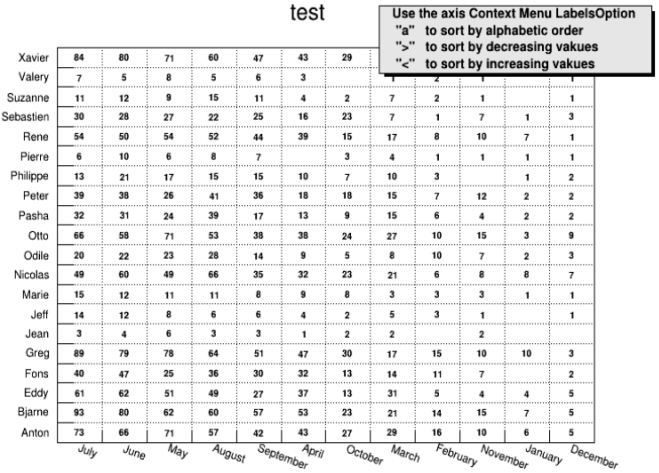
graphs/labels1.C



LabelsOption at 3rd of the pop menu

```
void labels1()
{
   const Int_t nx = 20;
   char *people[nx] = {"Jean","Pierre","Marie","Odile",
      "Sebastien","Fons","Rene","Nicolas","Xavier","Greg",
      "Bjarne","Anton","Otto","Eddy","Peter","Pasha",
      "Philippe","Suzanne","Jeff","Valery"};
   TCanvas *c1 = new TCanvas("c1","demo bin labels",
      10,10,900,500);
   c1->SetGrid();
   c1->SetBottomMargin(0.15);
   TH1F *h = new TH1F("h","test",nx,0,nx);
   h->SetFillColor(38);
   for (Int_t i=0;i<5000;i++) {
      h->Fill(gRandom->Gaus(0.5*nx,0.2*nx));
   }
   h->SetStats(0);
   for (i=1;i<=nx;i++) {
      h->GetXaxis()->SetBinLabel(i,people[i-1]);
   }
   h->Draw();
```

graphs/labels2.C

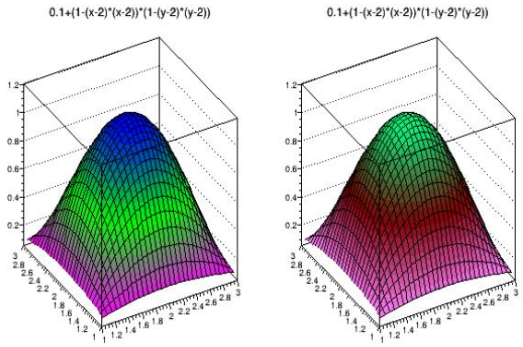



```
const Int_t ny = 20;
char *month[nx]  = {"January","February","March","April",
   "May","June","July","August","September","October",
   "November","December"};
char *people[ny] = {"Jean","Pierre","Marie","Odile",
   "Sebastien","Fons","Rene","Nicolas","Xavier","Greg",
   "Bjarne","Anton","Otto","Eddy","Peter","Pasha",
   "Philippe","Suzanne","Jeff","Valery"};
TCanvas *c1 = new TCanvas("c1","demo bin labels",
   10,10,800,800);
c1->SetGrid();
c1->SetLeftMargin(0.15);
c1->SetBottomMargin(0.15);
TH2F *h = new TH2F("h","test",nx,0,nx,ny,0,ny);
for (Int_t i=0;i<5000;i++) {
   h->Fill(gRandom->Gaus(0.5*nx,0.2*nx),
      gRandom->Gaus(0.5*ny,0.2*ny));
}
h->SetStats(0);
for (i=1;i<=nx;i++) h->GetXaxis()->SetBinLabel(i,month[i-1]);
for (i=1;i<=ny;i++) h->GetYaxis()->SetBinLabel(i,people[i-1]);
h->Draw("text");
```

graphs/multipalette.C

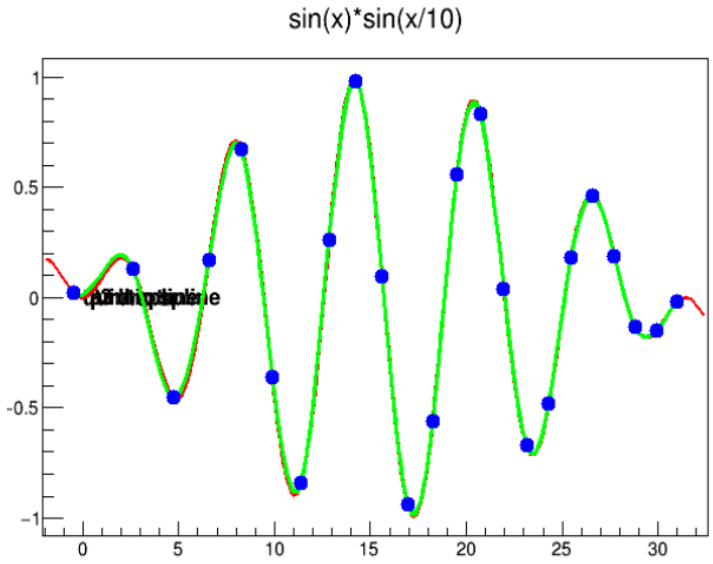
```
void Pal1()
{
   static Int_t  colors[50];
   static Bool_t initialized = kFALSE;

   Double_t Red[3]    = { 1.00, 0.00, 0.00};
   Double_t Green[3]  = { 0.00, 1.00, 0.00};
   Double_t Blue[3]   = { 1.00, 0.00, 1.00};
   Double_t Length[3] = { 0.00, 0.50, 1.00 };

   if(!initialized){
      Int_t FI = TColor::CreateGradientColorTable(3,Length,Red,Green,Blue,50);
      for (int i=0; i<50; i++) colors[i] = FI+i;
      initialized = kTRUE;
      return;
   }
   gStyle->SetPalette(50,colors);
}
```

Int_t CreateGradientColorTable(UInt_t Number, Double_t* Length, Double_t* Red, Double_t* Green, Double_t* Blue, UInt_t NColors)



In order to create a color table do the following:
Define the RGB Colors:
> UInt_t Number = 5;
> Double_t Red[5]   = { 0.00, 0.09, 0.18, 0.09, 0.00 };
> Double_t Green[5] = { 0.01, 0.02, 0.39, 0.68, 0.97 };
> Double_t Blue[5] = { 0.17, 0.39, 0.62, 0.79, 0.97 };
Define the length of the (color)-interval between this points
> Double_t Stops[5] = { 0.00, 0.34, 0.61, 0.84, 1.00 };
i.e. the color interval between Color 2 and Color 3 is
0.79 - 0.62 => 17 % of the total palette area between these colors

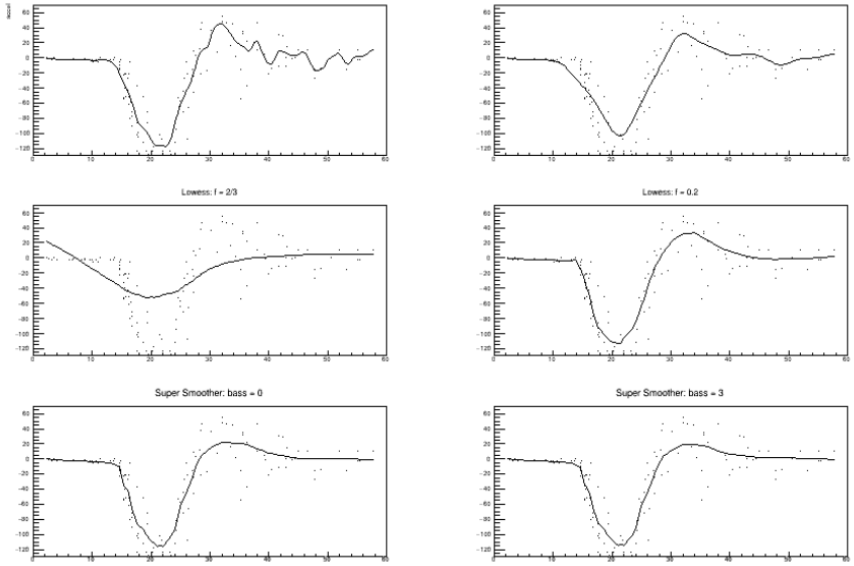
graphs/splines.C




```
// Define the original function
TF1 *f=new TF1("f","sin(x)*sin(x/10)",
               a-0.05*(b-a),b+0.05*(b-a));

  ........

// Evaluate fifth spline coefficients
Double_t eps=(b-a)*1.e-5;
if(spline5) delete spline5;
spline5 = new TSpline5("Test",xx,f,nnp,"b1e1b2e2",
                       f->Derivative(a),f->Derivative(b),
                       (f->Derivative(a+eps)-f->Derivative(a))/eps,
                       (f->Derivative(b)-f->Derivative(b-eps))/eps);

  ........

// Evaluate third spline coefficients
if(spline3) delete spline3;
spline3 = new TSpline3("Test",xx,yy,nnp,"b1e1",
                       f->Derivative(a),f->Derivative(b))
```

graphs/motorcycle.C

```
// Macro to test scatterplot smoothers: ksmooth, lowess, supsmu
// as described in:
//    Modern Applied Statistics with S-Plus, 3rd Edition
//    W.N. Venables and B.D. Ripley
//    Chapter 9: Smooth Regression, Figure 9.1
//
// Example is a set of data on 133 observations of acceleration against time
// for a simulated motorcycle accident, taken from Silverman (1985).
```



siguang@pku.edu.cn

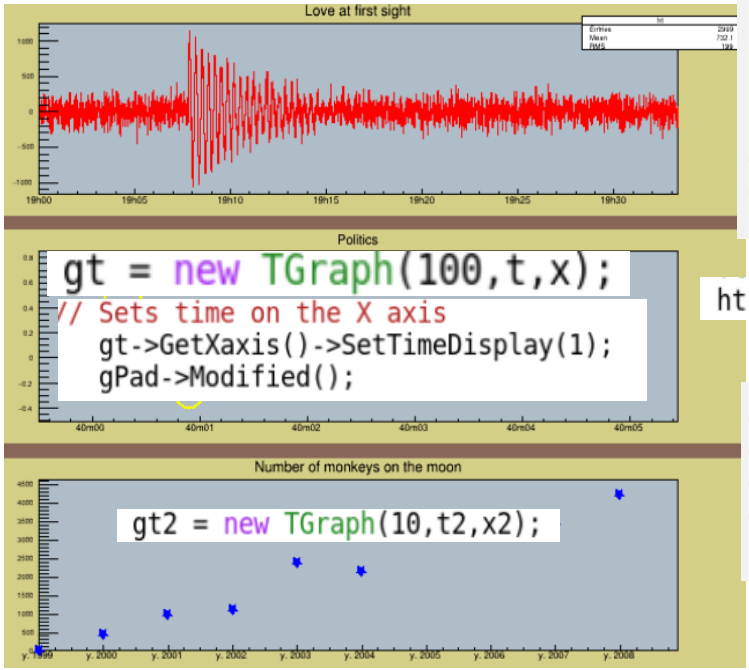
```
TCanvas *vC1;
TGraph *grin, *grout;

void DrawSmooth(Int_t pad, const char *title, const char *xt, const char *yt)
{
   vC1->cd(pad);
   TH1F *vFrame = gPad->DrawFrame(0,-130,60,70);
   vFrame->SetTitle(title);
   vFrame->SetTitleSize(0.2);
   vFrame->SetXTitle(xt);
   vFrame->SetYTitle(yt);
   grin->Draw("P");
   grout->DrawClone("LPX");
}
```

```
// data taken from R library MASS: mcycle.txt
   TString dir = gSystem->UnixPathName(gInterpreter->GetCurrentMacroName());
   dir.ReplaceAll("motorcycle.C","");
   dir.ReplaceAll("/./","/");

// read file and add to fit object
   Double_t *x = new Double_t[133];
   Double_t *y = new Double_t[133];
   Double_t vX, vY;
   Int_t vNData = 0;
   ifstream vInput;
   vInput.open(Form("%smotorcycle.dat",dir.Data()));
   while (1) {
      vInput >> vX >> vY;
      if (!vInput.good()) break;
      x[vNData] = vX;
      y[vNData] = vY;
      vNData++;
   }//while
   vInput.close();
```

```
grin = new TGraph(vNData,x,y);

vC1 = new TCanvas("vC1","Smooth Regression",200,10,900,700);
vC1->Divide(2,3);

Kernel Smoother
create new kernel smoother and smooth data with bandwidth = 2.0
TGraphSmooth *gs = new TGraphSmooth("normal");
grout = gs->SmoothKern(grin,"normal",2.0);
DrawSmooth(1,"Kernel Smoother: bandwidth = 2.0","times","accel");

redraw ksmooth with bandwidth = 5.0
grout = gs->SmoothKern(grin,"normal",5.0);
DrawSmooth(2,"Kernel Smoother: bandwidth = 5.0","","");

Lowess Smoother
create new lowess smoother and smooth data with fraction f = 2/3
grout = gs->SmoothLowess(grin,"",0.67);
DrawSmooth(3,"Lowess: f = 2/3","","");

redraw lowess with fraction f = 0.2
grout = gs->SmoothLowess(grin,"",0.2);
DrawSmooth(4,"Lowess: f = 0.2","","");
```

```
// Super Smoother
// create new super smoother and smooth data with default bass = 0 and span = 0
   grout = gs->SmoothSuper(grin,"",0,0);
   DrawSmooth(5,"Super Smoother: bass = 0","","");

// redraw supsmu with bass = 3 (smoother curve)
   grout = gs->SmoothSuper(grin,"",3);
   DrawSmooth(6,"Super Smoother: bass = 3","","");
```

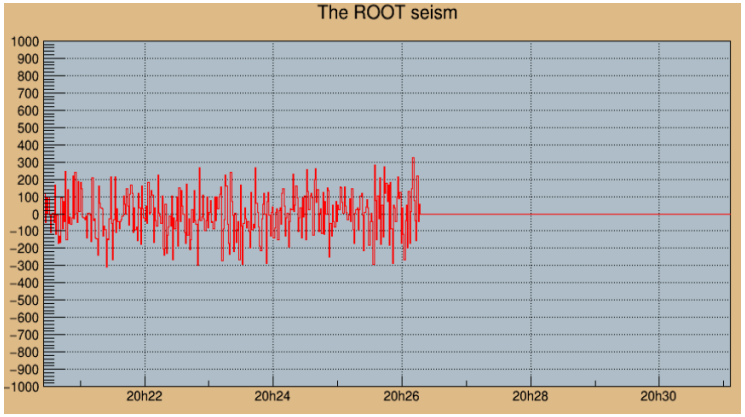
graphs/timeonaxis.C




```
   time_t script_time;
   script_time = time(0);
   script_time = 3600*(int)(script_time/3600);

// The time offset is the one that will be used by all graphs.
// If one changes it, it will be changed even on the graphs already define
   gStyle->SetTimeOffset(script_time);

   .......
ht = new TH1F("ht","Love at first sight",3000,0.,2000.);
   .......
// Sets time on the X axis
// The time used is the one set as time offset added to the value
// of the axis. This is converted into day/month/year hour:min:sec and
// a reasonnable tick interval value is chosen.
   ht->GetXaxis()->SetTimeDisplay(1);
```

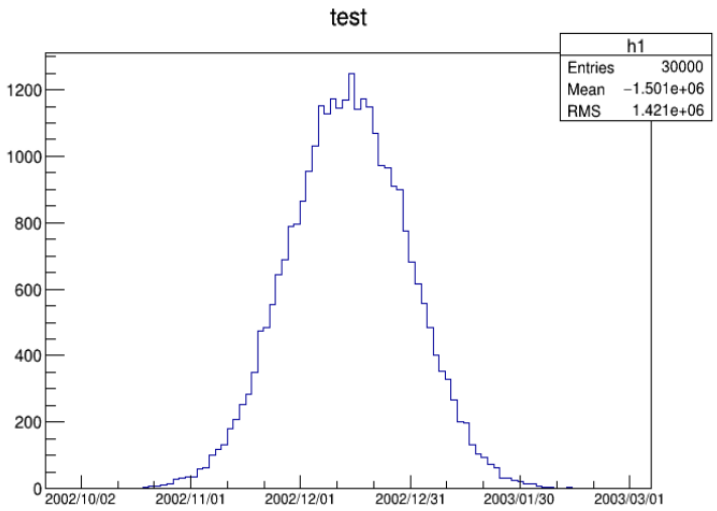
graphs/seism.C

```
TStopwatch sw; sw.Start();
//set time offset
TDatime dtime;
gStyle->SetTimeOffset(dtime.Convert());

Float_t bintime = 1; //one bin = 1 second. change it to set the time scale
TH1F *ht = new TH1F("ht","The ROOT seism",10,0,10*bintime);
Float_t signal = 1000;
ht->SetMaximum( signal);
ht->SetMinimum(-signal);
ht->SetStats(0);
ht->SetLineColor(2);
ht->GetXaxis()->SetTimeDisplay(1);
ht->GetYaxis()->SetNdivisions(520);
ht->Draw();

for (Int_t i=1;i<2300;i++) {
   //======= Build a signal : noisy damped sine ======
   Float_t noise  = gRandom->Gaus(0,120);
   if (i > 700) noise += signal*sin((i-700.)*6.28/30)*exp((700.-i)/300.);
   ht->SetBinContent(i,noise);
   c1->Modified();
   c1->Update();
   gSystem->ProcessEvents(); //canvas can be edited during the loop
}
```



```
//
// One can choose a different time format than the one chosen by default
// The time format is the same as the one of the C strftime() function
// It's a string containing the following formats :
//    for date :
//      %a abbreviated weekday name
//      %b abbreviated month name
//      %d day of the month (01-31)
//      %m month (01-12)
//      %y year without century
//      %Y year with century
//
//    for time :
//      %H hour (24-hour clock)
//      %I hour (12-hour clock)
//      %p local equivalent of AM or PM
//      %M minute (00-59)
//      %S seconds (00-61)
//      %% %
// The other characters are output as is.

   gt2->GetXaxis()->SetTimeFormat("y. %Y %F2000-01-01 00:00:00");
   gPad->Modified();
```

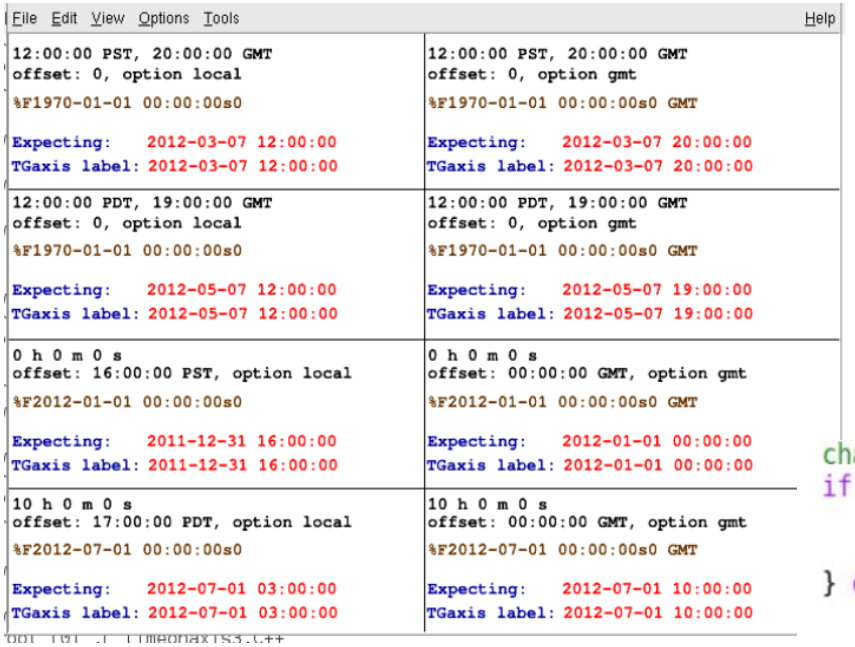
graphs/timeonaxis2.C




```
void timeonaxis2() {
   // Define the time offset as 2003, January 1st
   //Author: Olivier Couet

   TDatime T0(2003,01,01,00,00,00);
   int X0 = T0.Convert();
   gStyle->SetTimeOffset(X0);

   // Define the lowest histogram limit as 2002, September 23rd
   TDatime T1(2002,09,23,00,00,00);
   int X1 = T1.Convert()-X0;

   // Define the highest histogram limit as 2003, March 7th
   TDatime T2(2003,03,07,00,00,00);
   int X2 = T2.Convert(1)-X0;

   TH1F * h1 = new TH1F("h1","test",100,X1,X2);

   TRandom r;
   for (Int_t i=0;i<30000;i++) {
      Double_t noise = r.Gaus(0.5*(X1+X2),0.1*(X2-X1));
      h1->Fill(noise);
   }
```

```
h1->GetXaxis()->SetTimeDisplay(1);
h1->GetXaxis()->SetLabelSize(0.03);
h1->GetXaxis()->SetTimeFormat("%Y\/%m\/%d");
h1->Draw();
```

graphs/timeonaxis3.C

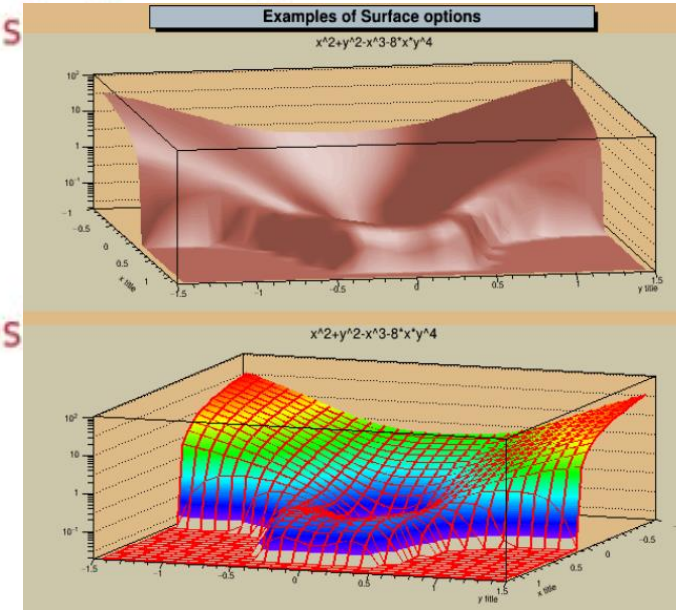



```
time_t offset[] = {0,                    0, 1325376000, 1341100800};
time_t t[]      = {1331150400, 1336417200,          0, 36000};

TGaxis* ga = new TGaxis (.4, .25, 5., .25, t[i], t[i] + 1,  1, "t");
ga->SetTimeFormat("TGaxis label: #color[2]{%Y-%m-%d %H:%M:%S}");
ga->SetLabelFont(102);
ga->SetLabelColor(kBlue+2);

ga->SetTimeOffset(offset[i], opt);
ga->SetLabelOffset(0.04*f);
ga->SetLabelSize(0.07*f);
ga->SetLineColor(0);
ga->Draw();
```

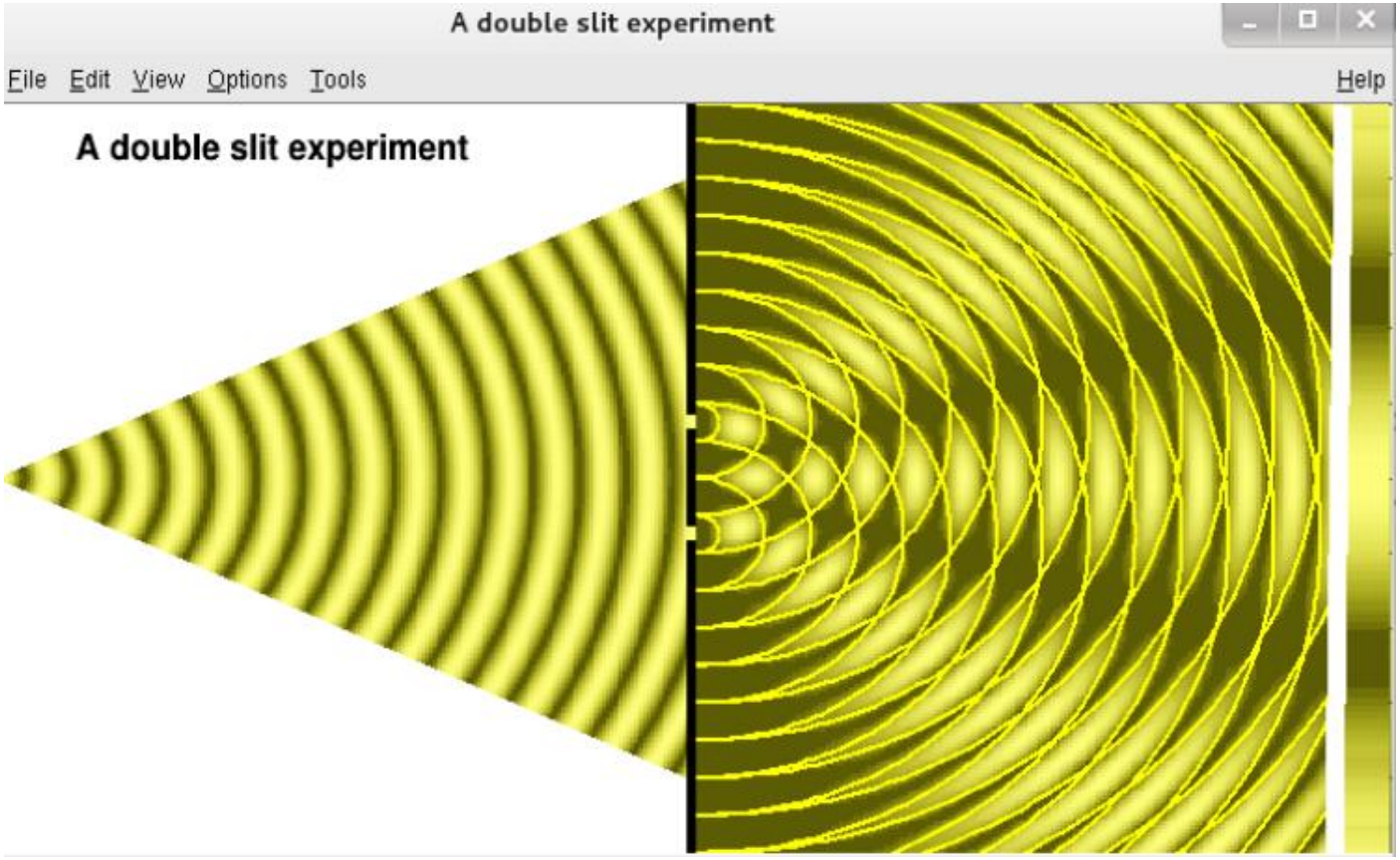
```
char buf[256];
if (offset[i] < t[i]) {
   sprintf(buf, "#splitline{%s, %s}{offset: %ld, option %s}",
           stime(t+i).Data(), stime(t+i, true).Data(), offset[i], opt);
} else {
   int h = t[i] / 3600;
   int m = (t[i] - 3600 * h) / 60 ;
   int s = (t[i] - h * 3600 - m * 60);
   sprintf(buf, "#splitline{%d h %d m %d s}{offset: %s, option %s}",
           h, m, s, stime(offset + i, gmt).Data(), opt);
}
tex1->DrawLatex(.01, .75, buf);
tex2->DrawLatex(.01, .50, offsettimeformat);
```

graphs/surfaces.C

```
// We generate a 2-D function
TF2 *f2 = new TF2("f2","x**2 + y**2 - x**3 -8*x*y**4",-1,1.2,-1.5,1.5);
f2->SetContour(48);
f2->SetFillColor(45);

// Draw this function in pad1 with Gouraud s
pad1->cd();
pad1->SetPhi(-80);
pad1->SetLogz();
f2->Draw("surf4");

// Draw this function in pad2 with color mes
pad2->cd();
pad2->SetTheta(25);
pad2->SetPhi(-110);
pad2->SetLogz();
f2->Draw("surf1");
```



graphs/waves.C



```
TF2 * finter;


//_____________________________________________________________
Double_t interference( Double_t *x, Double_t *par)
{
   Double_t x_p2 = x[0] * x[0];
   Double_t d_2 = 0.5 * par[2];
   Double_t ym_p2 = (x[1] - d_2) * (x[1] - d_2);
   Double_t yp_p2 = (x[1] + d_2) * (x[1] + d_2);
   Double_t  tpi_l = TMath::Pi() /  par[1];
   Double_t amplitude = par[0] * (cos(tpi_l  * sqrt(x_p2 + ym_p2))
                      + par[3] * cos(tpi_l  * sqrt(x_p2 + yp_p2)));
    return amplitude * amplitude;
}
```

No need spend time

```
//_____________________________________________________________
Double_t result( Double_t *x, Double_t *par)
{
   Double_t xint[2];
   Double_t  maxintens = 0, xcur = 14;
   Double_t dlambda = 0.1 * par[1];
   for(Int_t i=0; i<10; i++){
      xint[0] = xcur;
      xint[1] = x[1];
      Double_t  intens = interference(xint, par);
      if(intens > maxintens) maxintens = intens;
      xcur -= dlambda;
   }
   return maxintens;
}
```

No need spend time

```
void waves( Double_t d = 3, Double_t lambda = 1, Double_t amp = 10)
{
   TCanvas *c1 = new TCanvas("waves", "A double slit experiment",
      300,40, 1004, 759);
   c1->Range(0, -10,  30, 10);
   c1->SetFillColor(0);
   TPad *pad = new TPad("pr","pr",  0.5, 0 , 1., 1);
   pad->Range(0, -10,  15, 10);
   pad->Draw();

   const Int_t colNum = 30;
   Int_t palette[colNum];
   for (Int_t i=0;i<colNum;i++) {
      TColor *color = new TColor(1001+i
      ,    pow(i/((colNum)*1.0),0.3)
      ,    pow(i/((colNum)*1.0),0.3)
      ,0.5*(i/((colNum)*1.0)),"");
      palette[i] = 1001+i;
      if(!color) break;
   }
   gStyle->SetPalette(colNum,palette);
```

TColor(Int_t color, Float_t r, Float_t g, Float_t b, const char* name = "", Float_t a = 1)

```
//______________________________________________________
void TStyle::SetPalette(Int_t ncolors, Int_t *colors)
{
   // see TColor::SetPalette

   TColor::SetPalette(ncolors,colors);
}
```

myObject.SetFillColor(kRed); myObject.SetFillColor(kYellow-10); myLine.SetLineColor(kMagenta+2);

void SetPalette(Int_t ncolors, Int_t* colors)

if ncolors <= 0 a default palette (see below) of 50 colors is defined.
the colors defined in this palette are OK for coloring pads, labels

if ncolors == 1 && colors == 0, then
a Pretty Palette with a Spectrum Violet->Red is created.
It is recommended to use this Pretty palette when drawing legos, surfaces or contours.

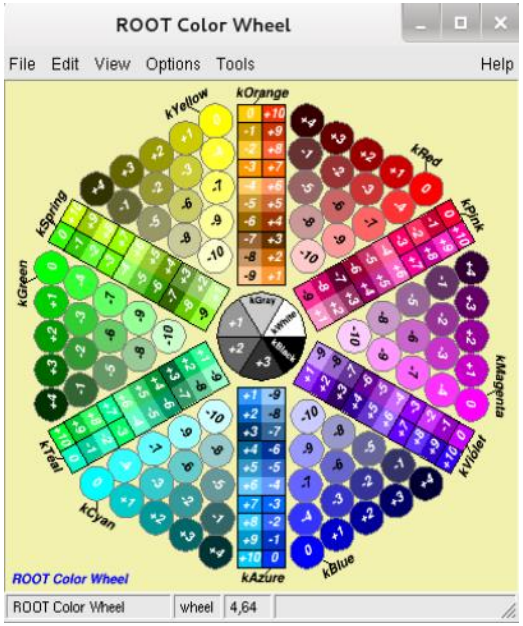
if ncolors > 50 and colors=0, the DeepSea palette is used.
(see TStyle::CreateGradientColorTable for more details)

if ncolors > 0 and colors = 0, the default palette is used with a maximum of ncolors.

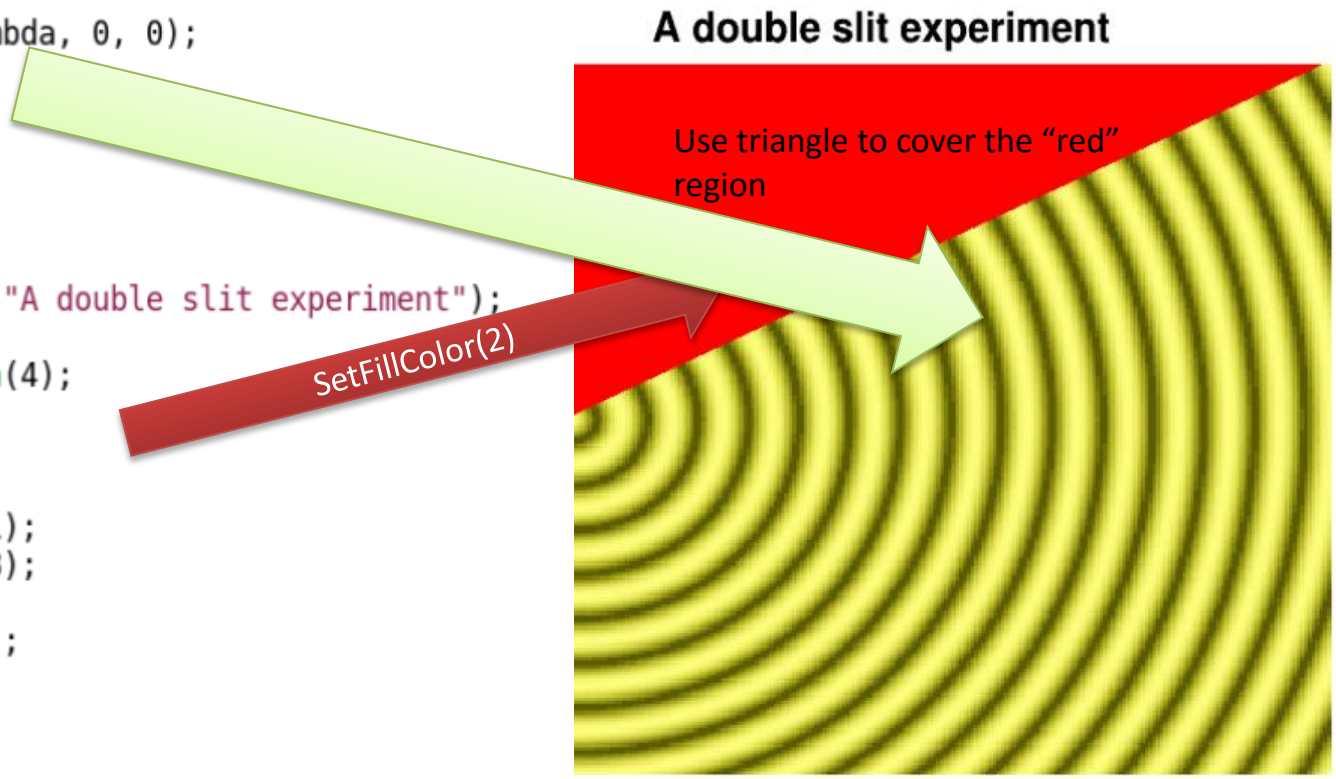
The default palette defines:
index 0->9 : grey colors from light to dark grey
index 10->19 : "brown" colors
index 20->29 : "blueish" colors
index 30->39 : "redish" colors
index 40->49 : basic colors

```
myObject.SetFillColor(kRed);
myObject.SetFillColor(kYellow-10);
myLine.SetLineColor(kMagenta+2);
```

```
root [4] TColorWheel w
root [5] w.Draw()
```

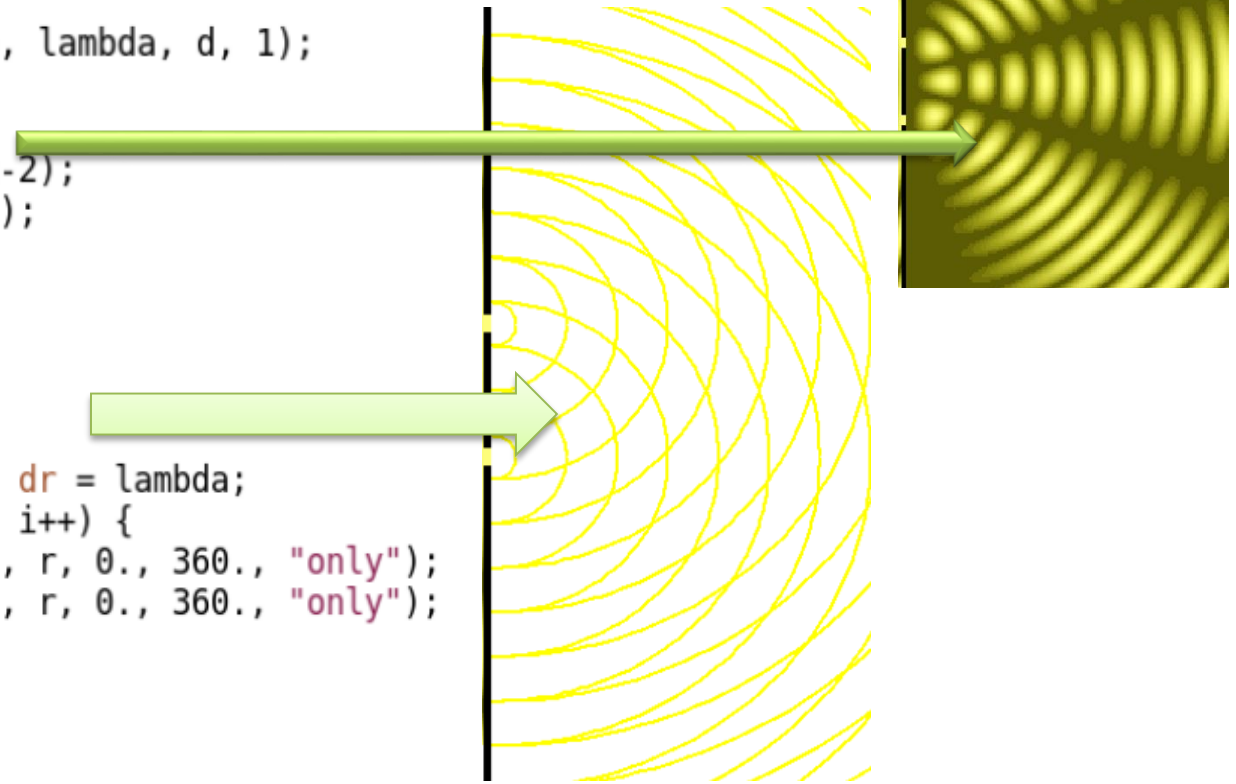


```
c1->cd();
TF2 * f0 = new TF2("ray_source",interference, 0.02, 15, -8, 8, 4);

f0->SetParameters(amp, lambda, 0, 0);
f0->SetNpx(200);
f0->SetNpy(200);
f0->SetContour(colNum-2);
f0->Draw("samecolz");

TLatex title;
title.DrawLatex(1.6, 8.5, "A double slit experiment");

TGraph *graph = new TGraph(4);
graph->SetFillColor(0);
graph->SetFillStyle(1001);
graph->SetLineWidth(0);
graph->SetPoint(0, 0., 0.1);
graph->SetPoint(1, 14.8, 8);
graph->SetPoint(2, 0, 8);
graph->SetPoint(3, 0, 0.1);
graph->Draw("F");
```

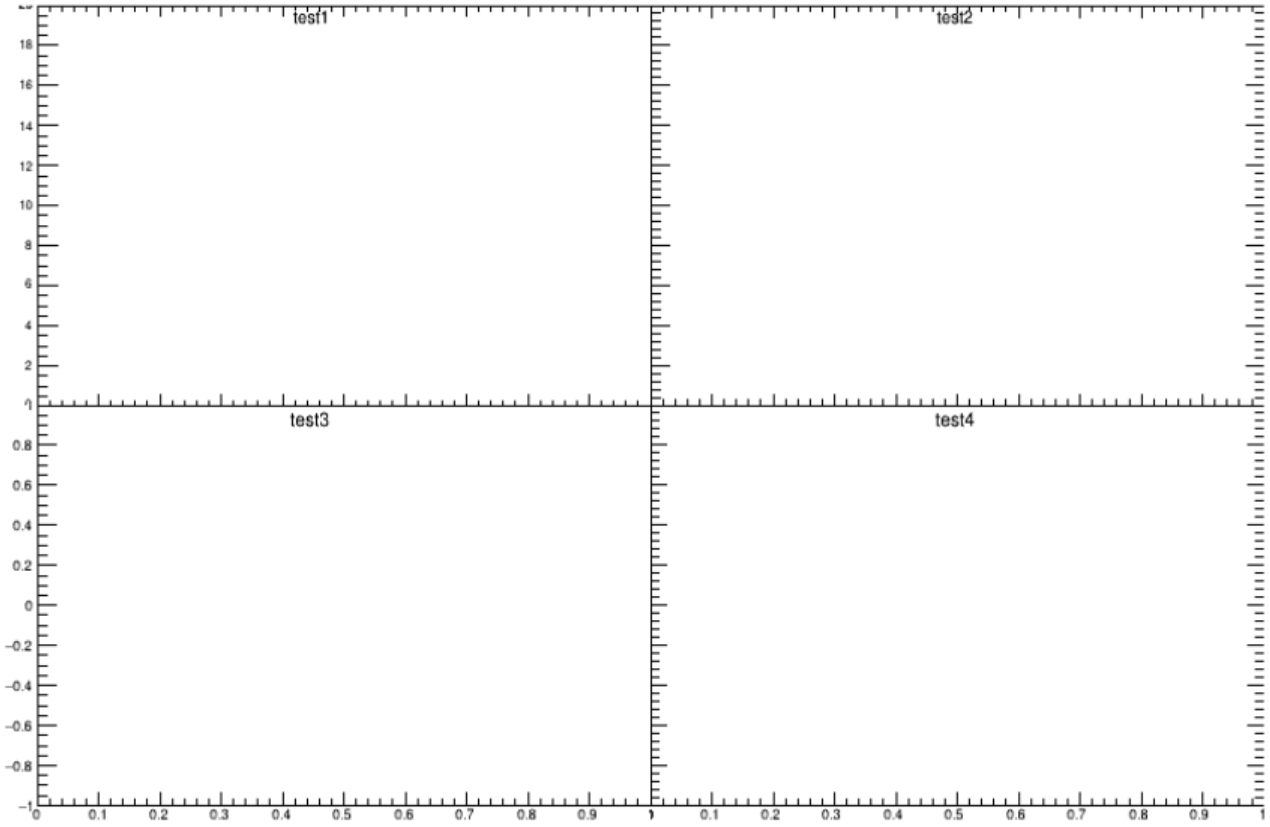


TF2 TF2(const char* name, void* fcn, Double_t xmin = 0, Double_t xmax = 1, Double_t ymin = 0, Double_t ymax = 1, Int_t npar = 0)

```
finter = new TF2("interference",interference, 0.01, 14, -10, 10, 4);

finter->SetParameters(amp, lambda, d, 1);
finter->SetNpx(200);
finter->SetNpy(200);
finter->SetContour(colNum-2);
finter->Draw("samecolorz");


TArc *arc = new TArc();;
arc->SetFillStyle(0);
arc->SetLineWidth(2);
arc->SetLineColor(5);
Float_t r = 0.5 * lambda, dr = lambda;
for (Int_t i = 0; i < 16; i++) {
   arc->DrawArc(0,  0.5*d, r, 0., 360., "only");
   arc->DrawArc(0, -0.5*d, r, 0., 360., "only");
   r += dr;
}
```



graphs/zones.C

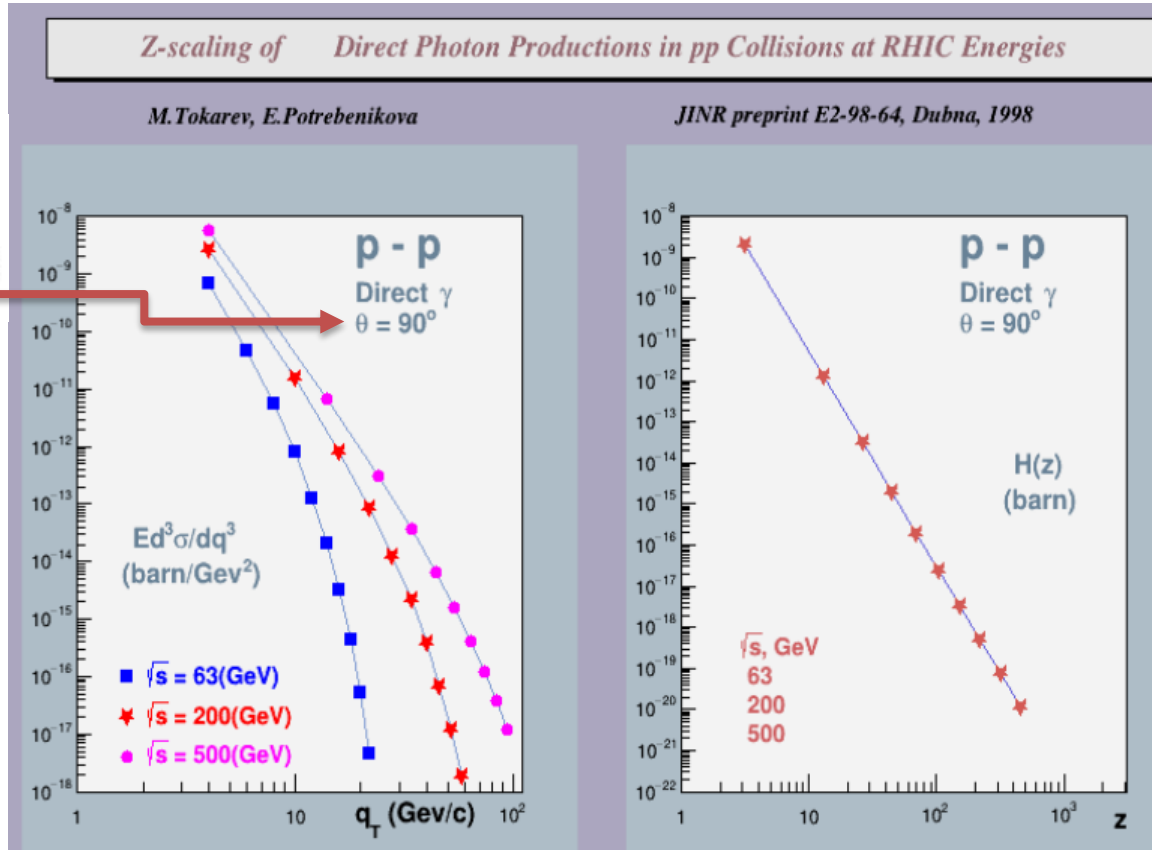


siguang@pku.edu.cn

```
TCanvas *c1 = new TCanvas("c1","multipads",900,700);
gStyle->SetOptStat(0);
c1->Divide(2,2,0,0);
TH2F *h1 = new TH2F("h1","test1",10,0,1,20,0,20);
TH2F *h2 = new TH2F("h2","test2",10,0,1,20,0,100);
TH2F *h3 = new TH2F("h3","test3",10,0,1,20,-1,1);
TH2F *h4 = new TH2F("h4","test4",10,0,1,20,0,1000);

c1->cd(1);
gPad->SetTickx(2);
h1->Draw();

c1->cd(2);
gPad->SetTickx(2);
gPad->SetTicky(2);
h2->GetYaxis()->SetLabelOffset(0.01);
h2->Draw();

c1->cd(3);
h3->Draw();

c1->cd(4);
gPad->SetTicky(2);
h4->Draw();
```

graphs/zdemo.C

```
t->SetTextSize(0.05);
t->DrawLatex(0.6,0.79,"Direct #gamma");
t->DrawLatex(0.6,0.75,"#theta = 90^{o}");

t->DrawLatex(0.20,0.45,"Ed^{3}#sigma/dq^{3}");
t->DrawLatex(0.18,0.40,"(barn/Gev^{2})");
```



Codes under $ROOTSYS/tutorials/hist

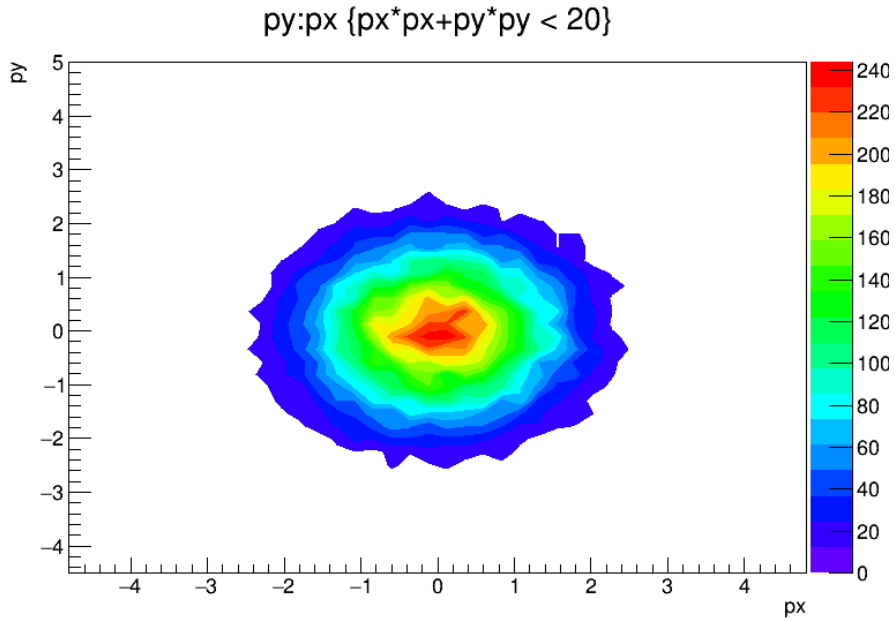
Useful functions of Histogram

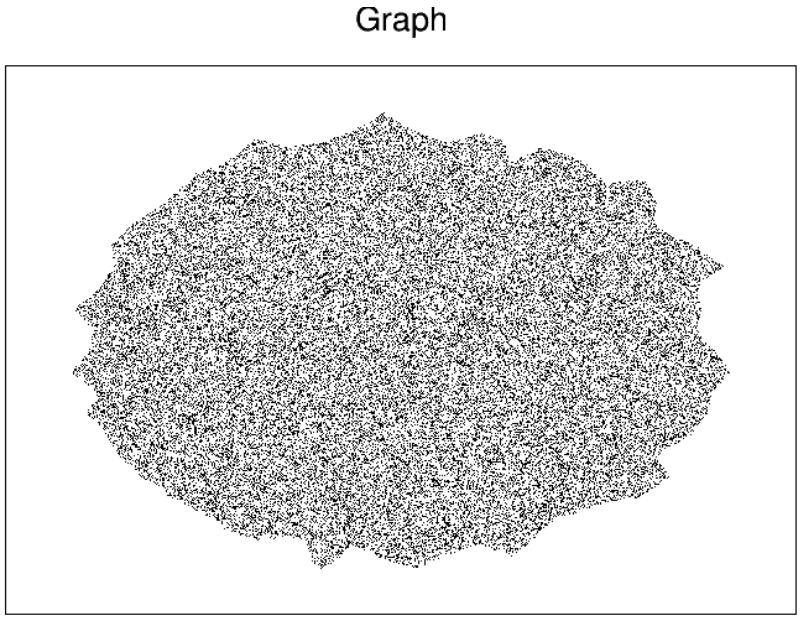
```
root [] h1->GetEntries()
(const Double_t)1.0000000000000000e+03
root [] h1->Integral()
(const Double_t)1.0000000000000000e+03
root [] h1->GetMean()
(const Double_t)1.1017279203592710e-02
root [] h1->GetMeanError()
(const Double_t)3.1831174486931387e-02
root [] h1->GetRMS()
(const Double_t)1.0065901197694480e+00
root [] h1->GetRMSError()
(const Double_t)2.2508039332841407e-02
```

H->Draw("options")
Options for 1D:

"AXIS"	Draw only axis
"AH"	Draw histogram, but not the axis labels and tick marks
"]["	When this option is selected the first and last vertical lines of the histogram are not drawn.
"B"	Bar chart option
"C"	Draw a smooth Curve througth the histogram bins
"E"	Draw error bars
"E0"	Draw error bars including bins with o contents
"E1"	Draw error bars with perpendicular lines at the edges
"E2"	Draw error bars with rectangles
"E3"	Draw a fill area througth the end points of the vertical error bars
"E4"	Draw a smoothed filled area through the end points of the error bars
"L"	Draw a line througth the bin contents
"P"	Draw current marker at each bin except empty bins
"P0"	Draw current marker at each bin including empty bins
"*H"	Draw histogram with a * at each bin
"LF2"	Draw histogram like with option "L" but with a fill area. Note that "L" draws also a fill area if the hist fillcolor is set but the fill area corresponds to the histogram contour.

/tutorials/hist/FirstContour.C





siguang@pku.edu.cn

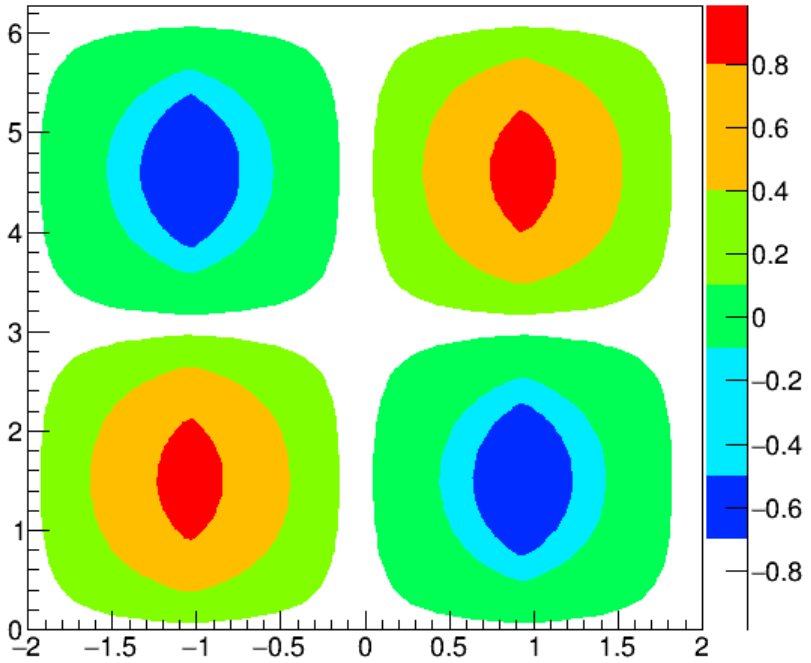
```
// make a contour plot and get the first contour in a TPolyMarker
void FirstContour()
{
   //this macro generates a color contour plot by selecting entries
   //from an ntuple file.
   //The TGraph object corresponding to the first contour line is
   //accessed and displayed into a separate canvas.
   //Author: Rene Brun

   TFile *file = TFile::Open("hsimple.root");
   if (!file) return;
   TTree *ntuple = (TTree*)file->Get("ntuple");

   TCanvas *c1 = new TCanvas("c1","Contours",10,10,800,600);
   gStyle->SetPalette(1);
   ntuple->Draw("py:px","px*px+py*py < 20", "contz,list");

   //we must call Update to force the canvas to be painted.  When
   //painting the contour plot, the list of contours is generated
   //and a reference to it added to the Root list of special objects
   c1->Update();
```

```
TCanvas *c2 = new TCanvas("c2","First contour",100,100,800,600);


TObjArray *contours =
   (TObjArray*)gROOT->GetListOfSpecials()->FindObject("contours");
if (!contours) return;
TList *lcontour1 = (TList*)contours->At(0);
if (!lcontour1) return;
TGraph *gc1 = (TGraph*)lcontour1->First();
if (!gc1) return;
if (gc1->GetN() < 10) return;
gc1->SetMarkerStyle(21);
gc1->Draw("alp");

//We make a TCutG object with the array obtained from this graph
TCutG *cutg = new TCutG("cutg",gc1->GetN(),gc1->GetX(),gc1->GetY());
```

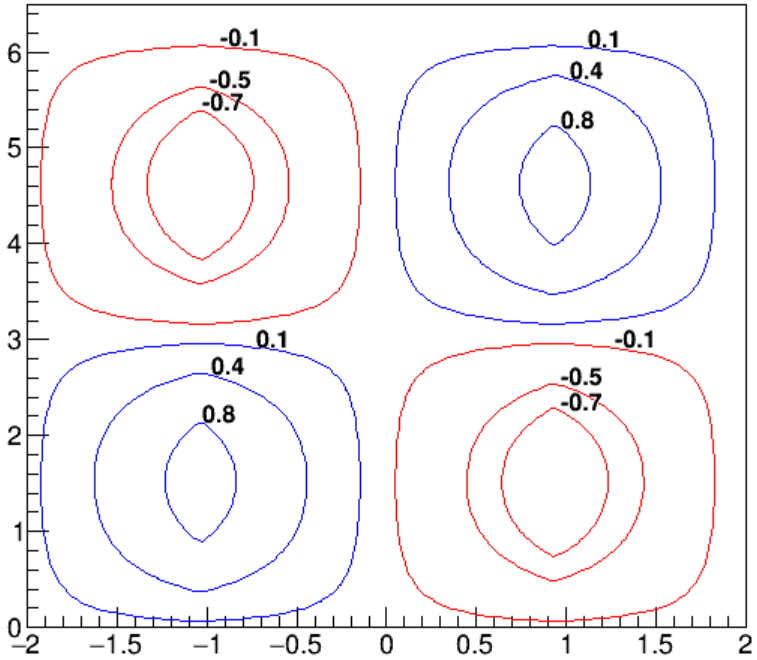
```
   //We make a TCutG object with the array obtained from this graph
   TCutG *cutg = new TCutG("cutg",gc1->GetN(),gc1->GetX(),gc1->GetY());

   //We create a polymarker object with npmax points.
   const Int_t npmax = 50000;
   TPolyMarker *pm = new TPolyMarker(npmax);
   Int_t np = 0;
   while(1) {
      Double_t x = -4 +8*gRandom->Rndm();
      Double_t y = -4 +8*gRandom->Rndm();
      if (cutg->IsInside(x,y)) {
         pm->SetPoint(np,x,y);
         np++;
         if (np == npmax) break;
      }
   }
   pm->Draw();
}
//--------------end of script contours.C
```

Tutorials/hist/ContourList.C

Histogram with negative and positive contents. Six contours are defined.
It is plotted with options CONT LIST to retrieve the contours points in TGraphs



Negative contours are returned first (highest to lowest). Positive contours are returned from lowest to highest. On this plot Negative contours are drawn in red and positive contours in blue.



siguang@pku.edu.cn

```
TCanvas *ContourList(){

   const Double_t PI = TMath::Pi();

   TCanvas* c = new TCanvas("c","Contour List",0,0,600,600);
   c->SetRightMargin(0.15);
   c->SetTopMargin(0.15);

   Int_t i, j, TotalConts;

   Int_t nZsamples   = 80;
   Int_t nPhiSamples = 80;

   Double_t HofZwavelength = 4.0;       // 4 meters
   Double_t dZ             =  HofZwavelength/(Double_t)(nZsamples - 1);
   Double_t dPhi           = 2*PI/(Double_t)(nPhiSamples - 1);

   TArrayD z(nZsamples);
   TArrayD HofZ(nZsamples);
   TArrayD phi(nPhiSamples);
```

```
   TArrayD FofPhi(nPhiSamples);

   // Discretized Z and Phi Values
   for ( i = 0; i < nZsamples; i++) {
      z[i] = (i)*dZ - HofZwavelength/2.0;
      HofZ[i] = SawTooth(z[i], HofZwavelength);
   }

   for(Int_t i=0; i < nPhiSamples; i++){
      phi[i] = (i)*dPhi;
      FofPhi[i] = sin(phi[i]);
   }

   // Create Histogram
   TH2D *HistStreamFn = new TH2D("HstreamFn",
   "#splitline{Histogram with negative and positive contents. Six contours are d
efined.}{It is plotted with options CONT LIST to retrieve the contours points in
 TGraphs}",
   nZsamples, z[0], z[nZsamples-1], nPhiSamples, phi[0], phi[nPhiSamples-1]);
```

```
// Load Histogram Data
for (Int_t i = 0; i < nZsamples; i++) {
   for(Int_t j = 0; j < nPhiSamples; j++){
      HistStreamFn->SetBinContent(i,j, HofZ[i]*FofPhi[j]);
   }
}

gStyle->SetPalette(1);
gStyle->SetOptStat(0);
gStyle->SetTitleW(0.99);
gStyle->SetTitleH(0.08);

Double_t contours[6];
contours[0] = -0.7;
contours[1] = -0.5;
contours[2] = -0.1;
contours[3] =  0.1;
contours[4] =  0.4;
contours[5] =  0.8;

HistStreamFn->SetContour(6, contours);
```

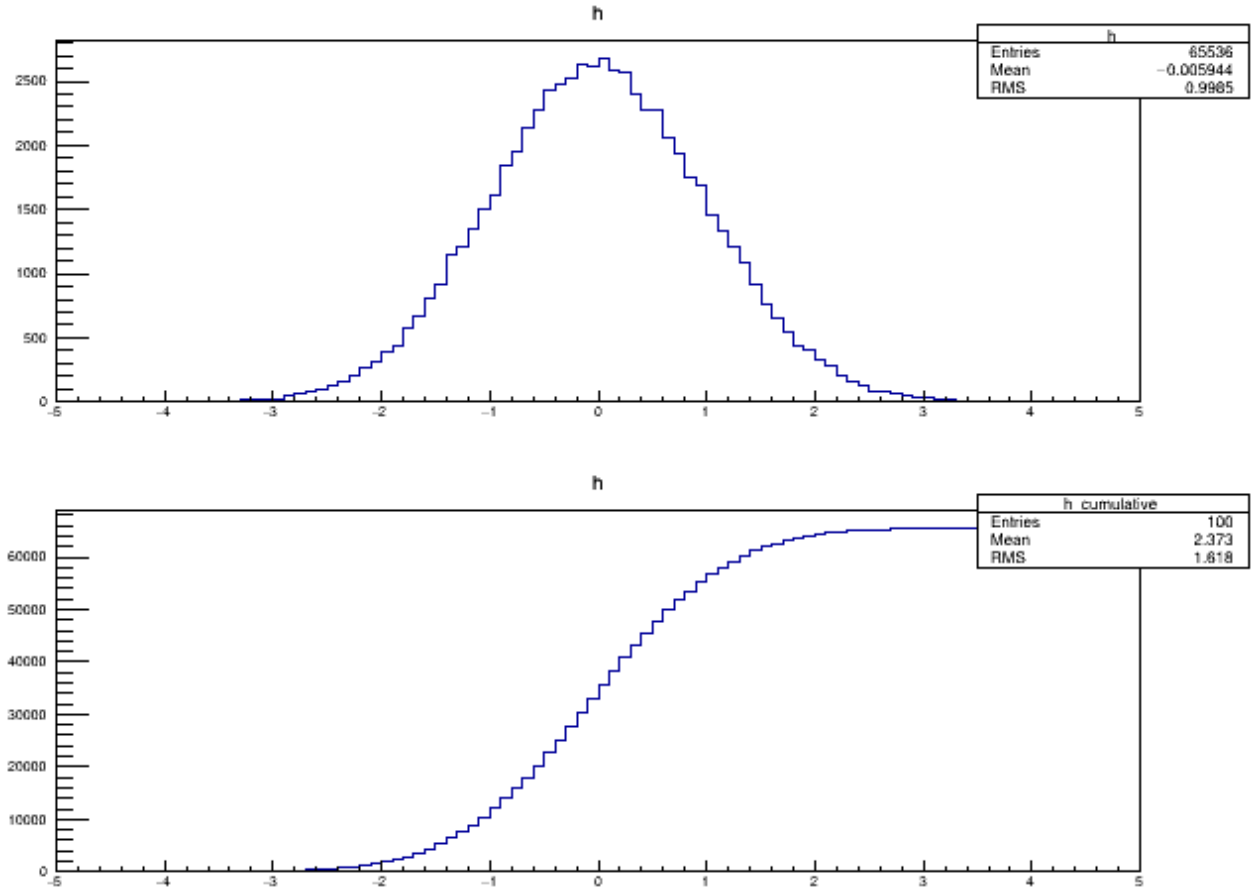
```
// Draw contours as filled regions, and Save points
HistStreamFn->Draw("CONT Z LIST");
c->Update(); // Needed to force the plotting and retrieve the contours in TGraphs

// Get Contours
TObjArray *conts = (TObjArray*)gROOT->GetListOfSpecials()->FindObject("contours");
TList* contLevel = NULL;
TGraph* curv     = NULL;
TGraph* gc       = NULL;

Int_t nGraphs    = 0;
Int_t TotalConts = 0;

if (conts == NULL){
   printf("*** No Contours Were Extracted!\n");
   TotalConts = 0;
   return;
} else {
   TotalConts = conts->GetSize();
}

printf("TotalConts = %d\n", TotalConts);
```

```
   for(i = 0; i < TotalConts; i++){
      contLevel = (TList*)conts->At(i);
      printf("Contour %d has %d Graphs\n", i, contLevel->GetSize());
      nGraphs += contLevel->GetSize();
   }

   nGraphs = 0;

   TCanvas* c1 = new TCanvas("c1","Contour List",610,0,600,600);
   c1->SetTopMargin(0.15);
   TH2F *hr = new TH2F("hr",
   "#splitline{Negative contours are returned first (highest to lowest). Positive contours are returned from}{lowest to highest. On
this plot Negative contours are drawn in red and positive contours in blue.}",
   2, -2, 2, 2, 0, 6.5);

   hr->Draw();
```

```
Double_t x0, y0, z0;
TLatex l;
l.SetTextSize(0.03);
char val[20];

for(i = 0; i < TotalConts; i++){
   contLevel = (TList*)conts->At(i);
   if (i<3) z0 = contours[2-i];
   else     z0 = contours[i];
   printf("Z-Level Passed in as:  Z = %f\n", z0);

   // Get first graph from list on curves on this level
   curv = (TGraph*)contLevel->First();

   for(j = 0; j < contLevel->GetSize(); j++){
      curv->GetPoint(0, x0, y0);
      if (z0<0) curv->SetLineColor(kRed);
      if (z0>0) curv->SetLineColor(kBlue);
      nGraphs ++;
      printf("\tGraph: %d  -- %d Elements\n", nGraphs,curv->GetN());
```

```
    // Get first graph from list on curves on this level
    curv = (TGraph*)contLevel->First();

    for(j = 0; j < contLevel->GetSize(); j++){
       curv->GetPoint(0, x0, y0);
       if (z0<0) curv->SetLineColor(kRed);
       if (z0>0) curv->SetLineColor(kBlue);
       nGraphs ++;
       printf("\tGraph: %d  -- %d Elements\n", nGraphs,curv->GetN());

       // Draw clones of the graphs to avoid deletions in case the 1st
       // pad is redrawn.
       gc = (TGraph*)curv->Clone();
       gc->Draw("C");

       sprintf(val,"%g",z0);
       l.DrawLatex(x0,y0,val);
       curv = (TGraph*)contLevel->After(curv); // Get Next graph
    }
}
c1->Update();
```

/tutorials/hist/cumulative.C



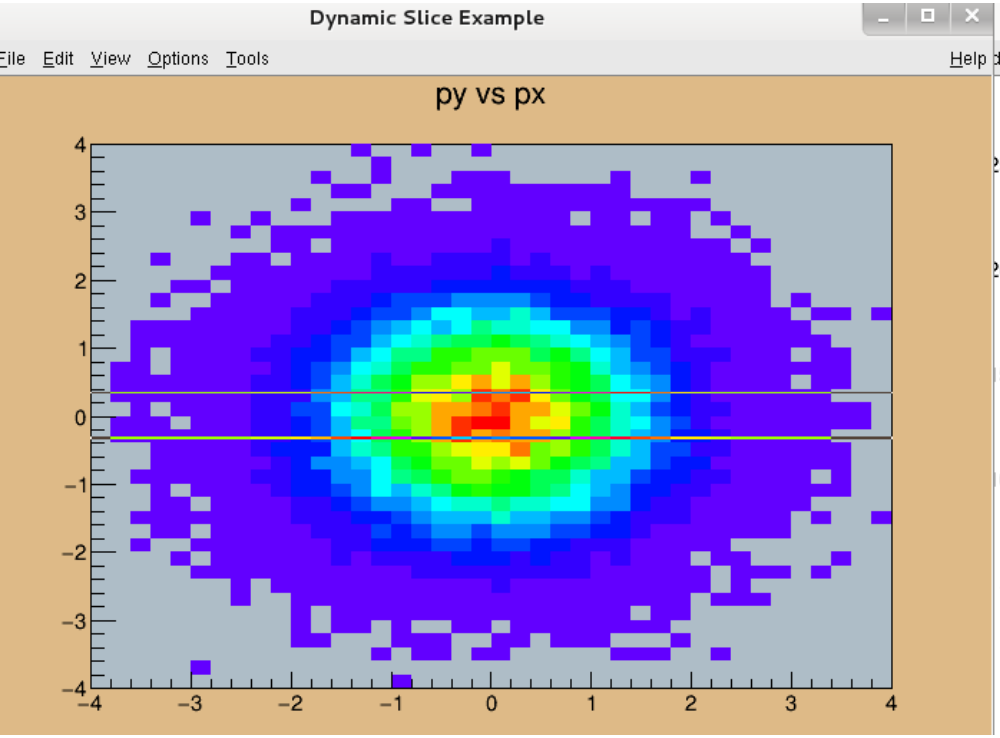
siguang@pku.edu.cn

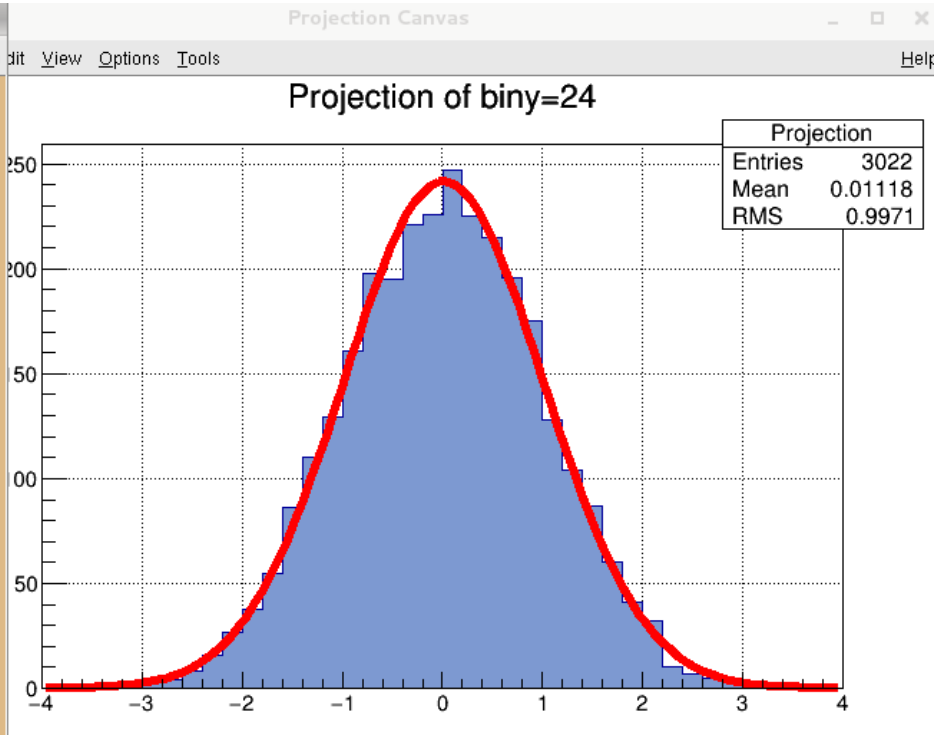
```
#include <cmath>

#include "TH1.h"
#include "TH1D.h"
#include "TCanvas.h"
#include "TRandom.h"

TCanvas* cumulative()
{
   TH1* h = new TH1D("h", "h", 100, -5., 5.);
   gRandom->SetSeed();
   h->FillRandom("gaus", 1u << 16);
   // get the cumulative of h
   TH1* hc = h->GetCumulative();
   // check that c has the "right" contents
   Double_t* integral = h->GetIntegral();
   for (Int_t i = 1; i <= hc->GetNbinsX(); ++i) {
      assert(std::abs(integral[i] * h->GetEntries() - hc->GetBinContent(i)) < 1e-7);
   }
   // draw histogram together with its cumulative distribution
   TCanvas* c = new TCanvas;
   c->Divide(1,2);
   c->cd(1);
```

```
root [0] 1u<<16
(const unsigned int)65536
root [1] 1u<<1
(const unsigned int)2
root [2] 1u<<2
(const unsigned int)4
root [3] 1u<<3
(const unsigned int)8
```

From 0 to 1

/tutorials/hist/DynamicSlice.C





siguang@pku.edu.cn

```
// Show the slice of a TH2 following the mouse position
void DynamicSlice()
{
   // Create a new canvas.
   c1 = new TCanvas("c1","Dynamic Slice Example",10,10,700,500);
   c1->SetFillColor(42);
   c1->SetFrameFillColor(33);

   //create a 2-d histogram, fill and draw it
   TH2F *hpxpy  = new TH2F("hpxpy","py vs px",40,-4,4,40,-4,4);
   hpxpy->SetStats(0);
   Double_t px,py;
   for (Int_t i = 0; i < 50000; i++) {
      gRandom->Rannor(px,py);
      hpxpy->Fill(px,py);
   }
   hpxpy->Draw("col");

   //Add a TExec object to the canvas
   c1->AddExec("dynamic","DynamicExec()");
}
```

```
void DynamicExec()
{
   // Example of function called when a mouse event occurs in a pad.
   // When moving the mouse in the canvas, a second canvas shows the
   // projection along X of the bin corresponding to the Y position
   // of the mouse. The resulting histogram is fitted with a gaussian.
   // A "dynamic" line shows the current bin position in Y.
   // This more elaborated example can be used as a starting point
   // to develop more powerful interactive applications exploiting CINT
   // as a development engine.
   //
   // Author:  Rene Brun

   TObject *select = gPad->GetSelected();
   if(!select) return;
   if (!select->InheritsFrom(TH2::Class())) {gPad->SetUniqueID(0); return;}
   TH2 *h = (TH2*)select;
   gPad->GetCanvas()->FeedbackMode(kTRUE);
```

```
//erase old position and draw a line at current position
int pyold = gPad->GetUniqueID();
int px = gPad->GetEventX();
int py = gPad->GetEventY();
float uxmin = gPad->GetUxmin();
float uxmax = gPad->GetUxmax();
int pxmin = gPad->XtoAbsPixel(uxmin);
int pxmax = gPad->XtoAbsPixel(uxmax);
if(pyold) gVirtualX->DrawLine(pxmin,pyold,pxmax,pyold);
gVirtualX->DrawLine(pxmin,py,pxmax,py);
gPad->SetUniqueID(py);
Float_t upy = gPad->AbsPixeltoY(py);
Float_t y = gPad->PadtoY(upy);
```

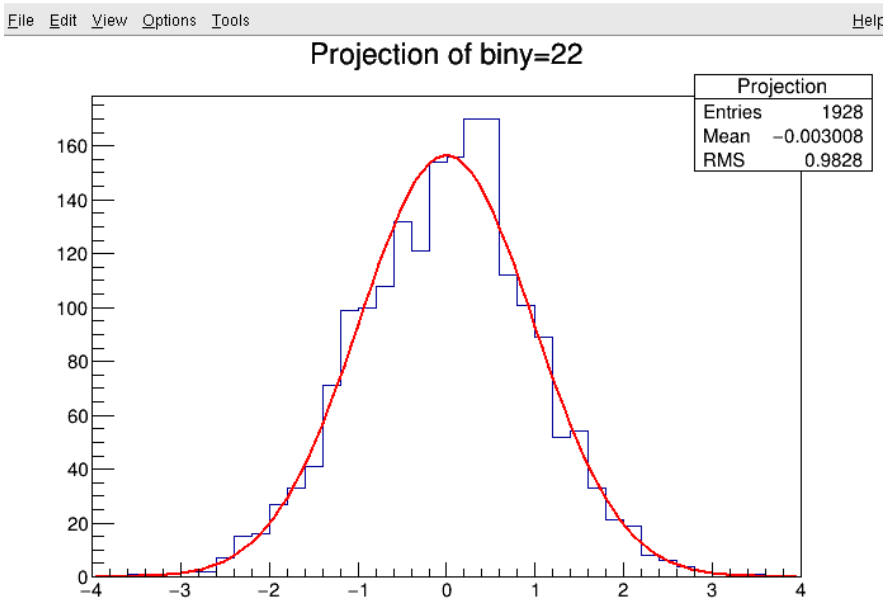
```
//create or set the new canvas c2
TVirtualPad *padsav = gPad;
TCanvas *c2 = (TCanvas*)gROOT->GetListOfCanvases()->FindObject("c2");
if(c2) delete c2->GetPrimitive("Projection");
else   c2 = new TCanvas("c2","Projection Canvas",710,10,700,500);
c2->SetGrid();
c2->cd();
```

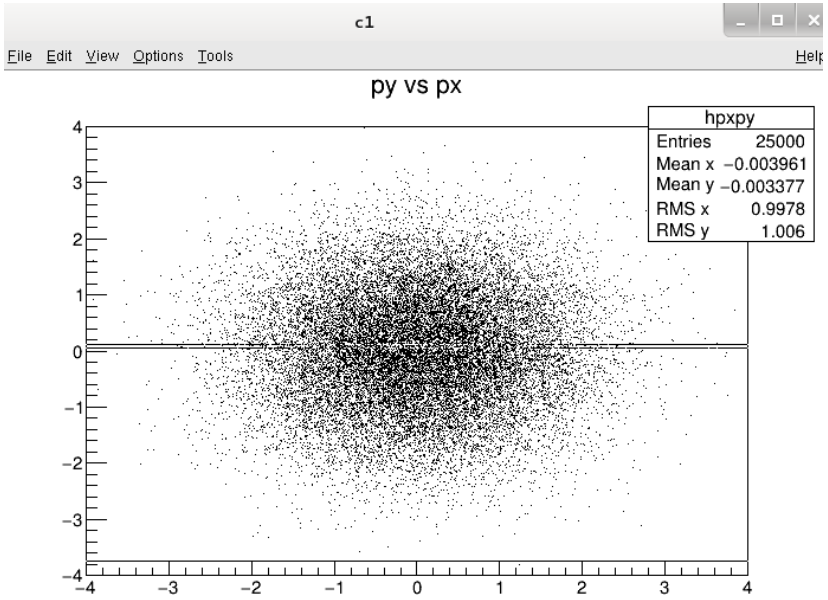
```
    //draw slice corresponding to mouse position
    Int_t biny = h->GetYaxis()->FindBin(y);
    TH1D *hp = h->ProjectionX("",biny,biny);
    hp->SetFillColor(38);
    char title[80];
    sprintf(title,"Projection of biny=%d",biny);
    hp->SetName("Projection");
    hp->SetTitle(title);
    hp->Fit("gaus","ql");
    hp->GetFunction("gaus")->SetLineColor(kRed);
    hp->GetFunction("gaus")->SetLineWidth(6);
    c2->Update();
    padsav->cd();
}
```

/tutorials/hist/exec2.C

```
root [0] TFile f("../hsimple.root");
root [1]  hpxpy.Draw();
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
root [2]  c1.AddExec("ex2",".x exec2.C");
```

If no hsimple.root
Run: /tutorials/hsimple.C





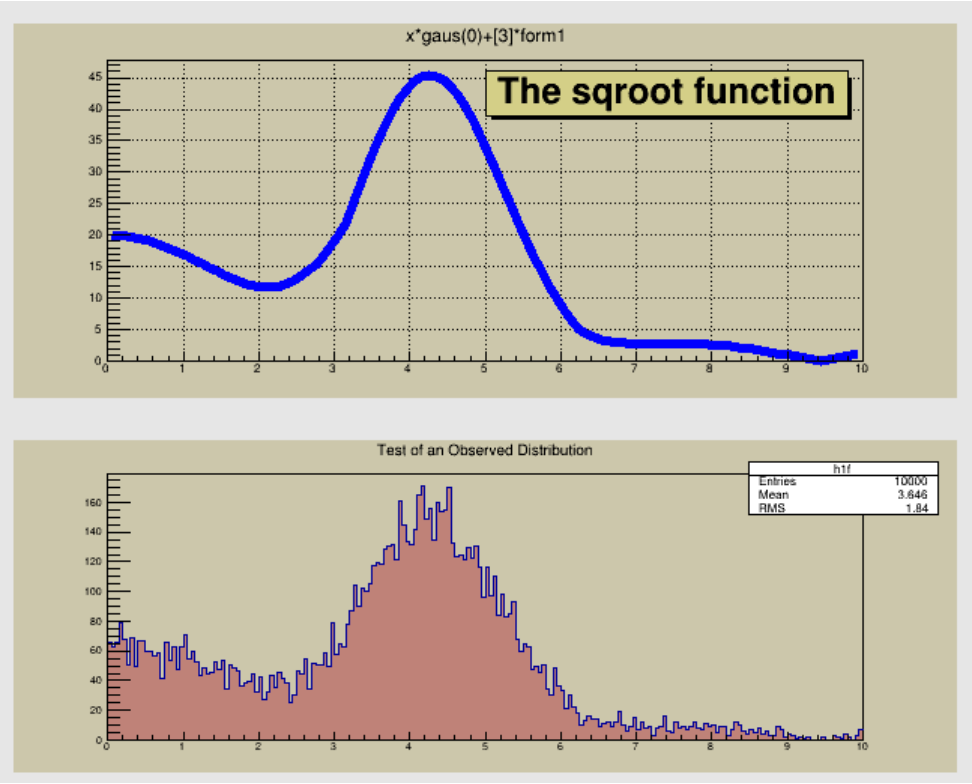
siguang@pku.edu.cn

```
// echo object at mouse position and show a graphics line
void exec2()
{
   //example of macro called when a mouse event occurs in a pad.
   // Example:
   // Root > TFile f("hsimple.root");
   // Root > hpxpy.Draw();
   // Root > c1.AddExec("ex2",".x exec2.C");
   // When moving the mouse in the canvas, a second canvas shows the
   // projection along X of the bin corresponding to the Y position
   // of the mouse. The resulting histogram is fitted with a gaussian.
   // A "dynamic" line shows the current bin position in Y.
   // This more elaborated example can be used as a starting point
   // to develop more powerful interactive applications exploiting CINT
   // as a development engine.
   //Author: Rene Brun

   TObject *select = gPad->GetSelected();
   if(!select) return;
   if (!select->InheritsFrom(TH2::Class())) {gPad->SetUniqueID(0); return;}
   gPad->GetCanvas()->FeedbackMode(kTRUE);
```

```
//erase old position and draw a line at current position
int pyold = gPad->GetUniqueID();
int px = gPad->GetEventX();
int py = gPad->GetEventY();
float uxmin = gPad->GetUxmin();
float uxmax = gPad->GetUxmax();
int pxmin = gPad->XtoAbsPixel(uxmin);
int pxmax = gPad->XtoAbsPixel(uxmax);
if(pyold) gVirtualX->DrawLine(pxmin,pyold,pxmax,pyold);
gVirtualX->DrawLine(pxmin,py,pxmax,py);
gPad->SetUniqueID(py);
Float_t upy = gPad->AbsPixeltoY(py);
Float_t y = gPad->PadtoY(upy);
```

```
   //create or set the new canvas c2
   TVirtualPad *padsav = gPad;
   TCanvas *c2 = (TCanvas*)gROOT->GetListOfCanvases()->FindObject("c2");
   if(c2) delete c2->GetPrimitive("Projection");
   else   c2 = new TCanvas("c2");
   c2->cd();

   //draw slice corresponding to mouse position
   TH2 *h = (TH2*)select;
   Int_t biny = h->GetYaxis()->FindBin(y);
   TH1D *hp = h->ProjectionX("",biny,biny);
   char title[80];
   sprintf(title,"Projection of biny=%d",biny);
   hp->SetName("Projection");
   hp->SetTitle(title);
   hp->Fit("gaus","ql");
   c2->Update();
   padsav->cd();
}
```

/tutorial/hist/fillrandom.C



siguang@pku.edu.cn

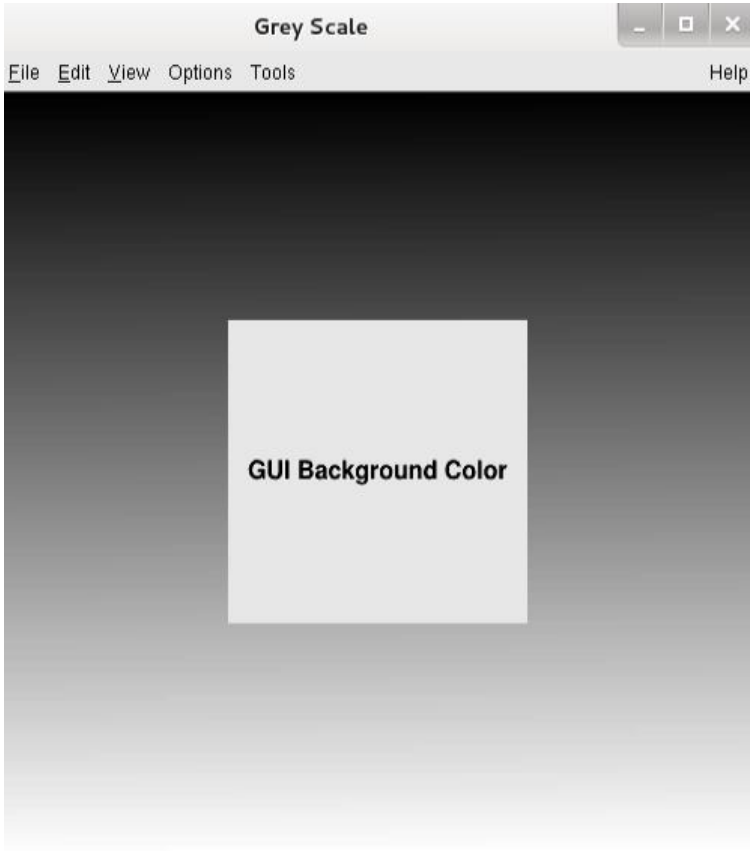
```
void fillrandom() {
   //Fill a 1-D histogram from a parametric function
   // To see the output of this macro, click begin_html <a href="gif/fillrandom.gif">here</a>. end_htm
   //Author: Rene Brun

   TCanvas *c1 = new TCanvas("c1","The FillRandom example",200,10,700,900);
   c1->SetFillColor(18);

   pad1 = new TPad("pad1","The pad with the function",0.05,0.50,0.95,0.95,21);
   pad2 = new TPad("pad2","The pad with the histogram",0.05,0.05,0.95,0.45,21);
   pad1->Draw();
   pad2->Draw();
   pad1->cd();

   gBenchmark->Start("fillrandom");
   //
   // A function (any dimension) or a formula may reference
   // an already defined formula
   //
   form1 = new TFormula("form1","abs(sin(x)/x)");
   sqroot = new TF1("sqroot","x*gaus(0) + [3]*form1",0,10);
   sqroot->SetParameters(10,4,1,20);
```

。。。。

```
gBenchmark->Start("fillrandom");
//
// A function (any dimension) or a formula may reference
// an already defined formula
//
form1 = new TFormula("form1","abs(sin(x)/x)");
sqroot = new TF1("sqroot","x*gaus(0) + [3]*form1",0,10);
sqroot->SetParameters(10,4,1,20);
pad1->SetGridx();
pad1->SetGridy();
pad1->GetFrame()->SetFillColor(42);
pad1->GetFrame()->SetBorderMode(-1);
pad1->GetFrame()->SetBorderSize(5);
sqroot->SetLineColor(4);
sqroot->SetLineWidth(6);
sqroot->Draw();
lfunction = new TPaveLabel(5,39,9.8,46,"The sqroot function");
lfunction->SetFillColor(41);
lfunction->Draw();
```

```
   // Create a one dimensional histogram (one float per bin)
   // and fill it following the distribution in function sqroot.
   //
   pad2->cd();
   pad2->GetFrame()->SetFillColor(42);
   pad2->GetFrame()->SetBorderMode(-1);
   pad2->GetFrame()->SetBorderSize(5);
   h1f = new TH1F("h1f","Test of an Observed Distribution",200,0,10);
   h1f->SetFillColor(45);
   h1f->FillRandom("sqroot",10000);
   h1f->Draw();
   c1->Update();
   //
   // Open a ROOT file and save the formula, function and histogram
   //
   TFile myfile("fillrandom.root","RECREATE");
   form1->Write();
   sqroot->Write();
   h1f->Write();
   gBenchmark->Show("fillrandom");
}
```

/tutorials/hist/greyscale.C



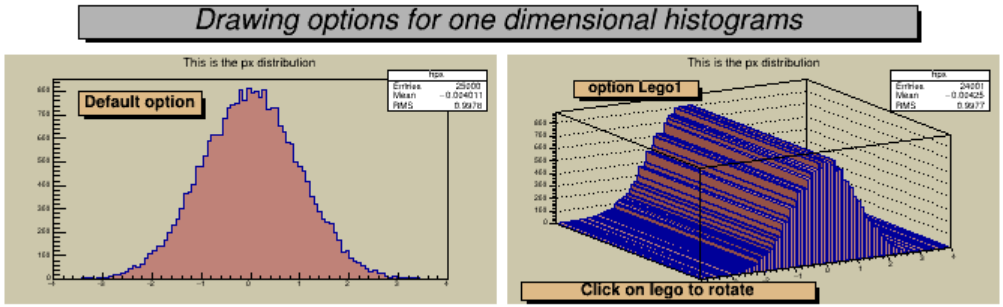
```
{
   // Create grey scale of nxn boxes.
   //Author: Olivier Couet

   if (gVirtualX) {
      Int_t n;
      gVirtualX->GetPlanes(n);
      if (n < 15) {
        printf("Not enough color planes to run this script, n=%d\n",n);
        // return;
      }
   }

   TCanvas *c = new TCanvas("grey", "Grey Scale", 500, 500);
   c->SetBorderMode(0);

   Int_t   n = 200;   // tunable parameter
   Float_t n1 = 1./n;
   for (int i = 0; i < n; i++) {
      for (int j = 0; j < n; j++) {
         TBox *b = new TBox(n1*j, n1*(n-1-i), n1*(j+1), n1*(n-i));
         Float_t grey = Float_t(i*n+j)/(n*n);
         b->SetFillColor(TColor::GetColor(grey, grey, grey));
```

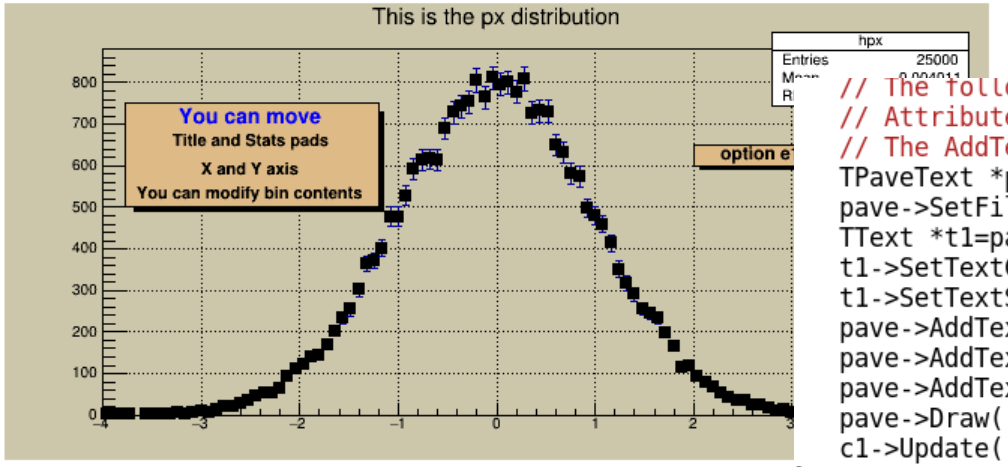
Even n=0, still can draw a nice plot. So comment out

grey: 0--1

```
            b->SetFillColor(TColor::GetColor(grey, grey, grey));
            b->Draw();
        }
    }
    TPad *p = new TPad("p","p",0.3, 0.3, 0.7,0.7);
    const char *guibackground = gEnv->GetValue("Gui.BackgroundColor", "");
    p->SetFillColor(TColor::GetColor(guibackground));
    p->Draw();
    p->cd();
    TText *t = new TText(0.5, 0.5, "GUI Background Color");
    t->SetTextAlign(22);
    t->SetTextSize(.09);
    t->Draw();

    c->SetEditable(kFALSE);
}
```

/tutorials/hist/h1draw.C






```
// The following illustrates how to add comments using a PaveText.
// Attributes of text/lines/boxes added to a PaveText can be modified.
// The AddText function returns a pointer to the added object.
TPaveText *pave = new TPaveText(-3.78,500,-1.2,750);
pave->SetFillColor(42);
TText *t1=pave->AddText("You can move");
t1->SetTextColor(4);
t1->SetTextSize(0.05);
pave->AddText("Title and Stats pads");
pave->AddText("X and Y axis");
pave->AddText("You can modify bin contents");
pave->Draw();
c1->Update();
```

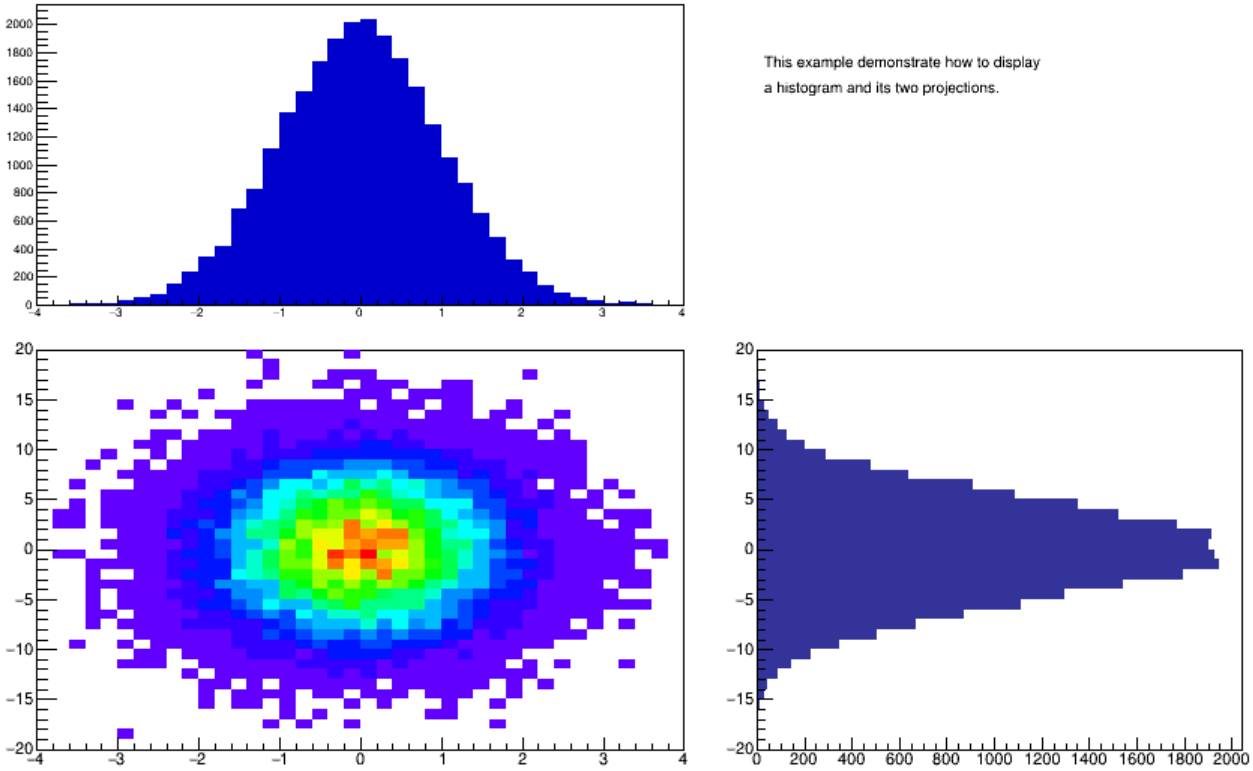
```
void h1draw()
{
   // We attach (or generate) the ROOT file in $ROOTSYS/tutorials/hsimple.root
   // or $PWD/hsimple.root
   // We draw one histogram in different formats
   //Author: Rene Brun
   TFile *example = TFile::Open("hsimple.root");
   if (!example) return;

   example->ls();
   TH1 *hpx = (TH1*)example->Get("hpx");

   TCanvas *c1 = new TCanvas("c1","Histogram Drawing Options",200,10,700,900);
   TPad *pad1 = new TPad("pad1",
      "The pad with the function",0.03,0.62,0.50,0.92,21);
   TPad *pad2 = new TPad("pad2",
      "The pad with the histogram",0.51,0.62,0.98,0.92,21);
   TPad *pad3 = new TPad("pad3",
      "The pad with the histogram",0.03,0.02,0.97,0.57,21);
   pad1->Draw();
```

.....

```
// Draw a global picture title
TPaveLabel *title = new TPaveLabel(0.1,0.94,0.9,0.98,
                   "Drawing options for one dimensional histograms");
title->SetFillColor(16);
title->SetTextFont(52);
title->Draw();

// Draw histogram hpx in first pad with the default option.
pad1->cd();
pad1->GetFrame()->SetFillColor(18);
hpx->SetFillColor(45);
hpx->DrawCopy();
TPaveLabel *label1 = new TPaveLabel(-3.5,700,-1,800,"Default option");
label1->SetFillColor(42);
label1->Draw();
```

```
pad2->cd();
hpx->DrawCopy("lego1");
TPaveLabel *label2 = new TPaveLabel(-0.72,0.74,-0.22,0.88,"option Lego1");
label2->SetFillColor(42);
label2->Draw();
TPaveLabel *label2a = new TPaveLabel(-0.93,-1.08,0.25,-0.92,
   "Click on lego to rotate");
label2a->SetFillColor(42);
label2a->Draw();

// Draw hpx with its errors and a marker.
pad3->cd();
pad3->SetGridx();
pad3->SetGridy();
pad3->GetFrame()->SetFillColor(18);
hpx->SetMarkerStyle(21);
hpx->Draw("e1p");
TPaveLabel *label3 = new TPaveLabel(2,600,3.5,650,"option e1p");
label3->SetFillColor(42);
```

/tutorials/hist/h2proj.C

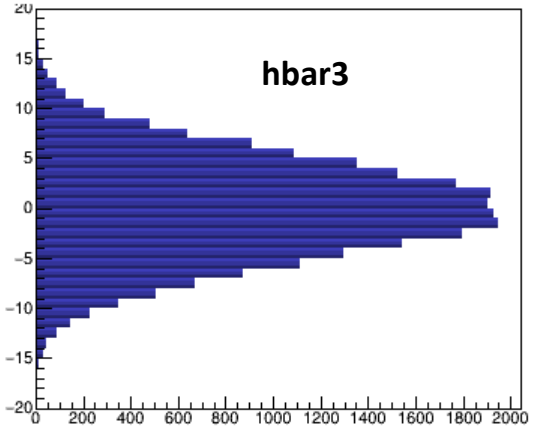


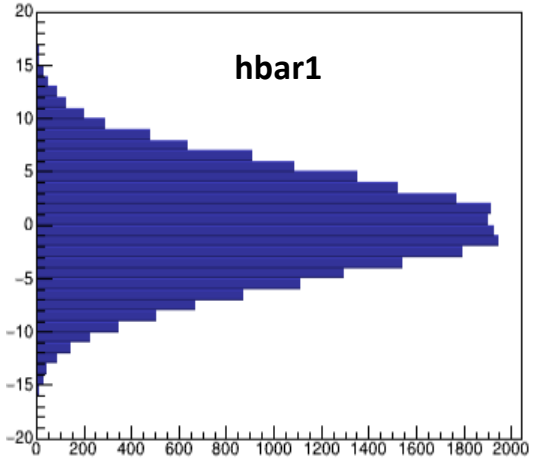
siguang@pku.edu.cn

```
{
   TCanvas *c1 = new TCanvas("c1", "c1",900,900);
   gStyle->SetOptStat(0);

   // Create the three pads
   TPad *center_pad = new TPad("center_pad", "center_pad",0.0,0.0,0.6,0.6);
   center_pad->Draw();

   right_pad = new TPad("right_pad", "right_pad",0.55,0.0,1.0,0.6);
   right_pad->Draw();

   top_pad = new TPad("top_pad", "top_pad",0.0,0.55,0.6,1.0);
   top_pad->Draw();

   // Create, fill and project a 2D histogram.
   TH2F *h2 = new TH2F("h2","",40,-4,4,40,-20,20);
   Float_t px, py;
   for (Int_t i = 0; i < 25000; i++) {
      gRandom->Rannor(px,py);
      h2->Fill(px,5*py);
   }
```

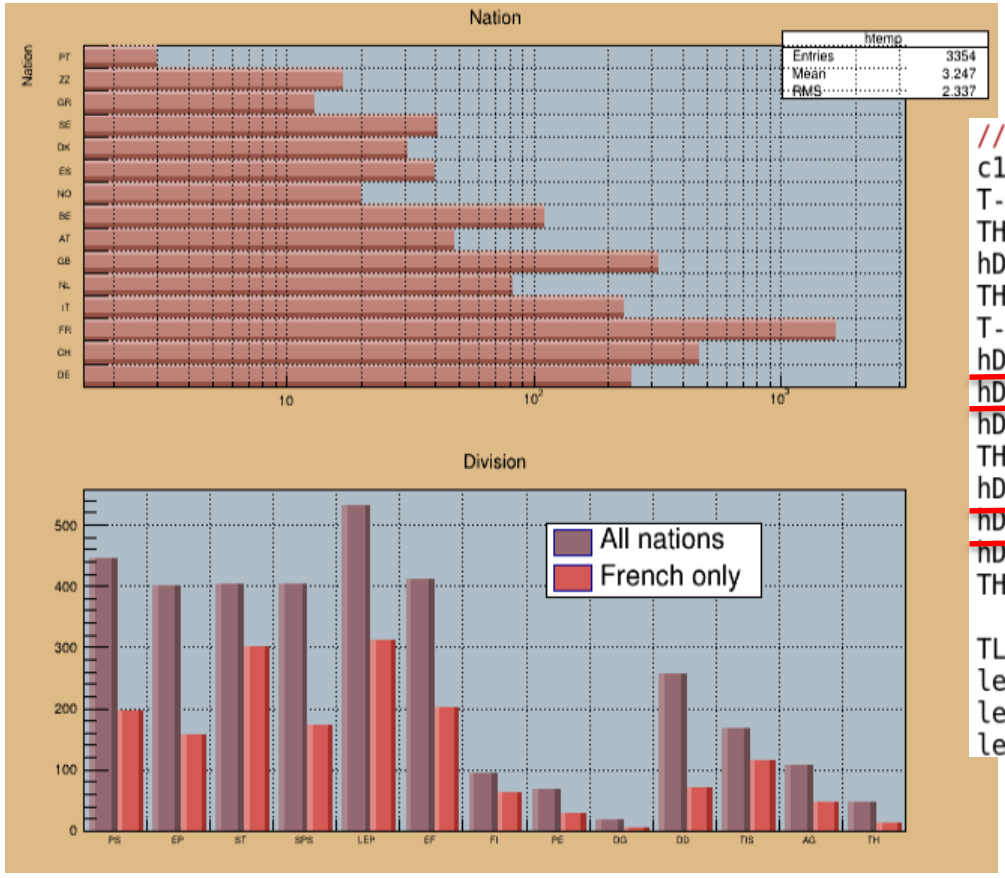
```
TH1D * projh2X = h2->ProjectionX();
TH1D * projh2Y = h2->ProjectionY();

// Drawing
center_pad->cd();
gStyle->SetPalette(1);
h2->Draw("COL");

top_pad->cd();
projh2X->SetFillColor(kBlue+1);
projh2X->Draw("bar");

right_pad->cd();
projh2Y->SetFillColor(kBlue-2);
projh2Y->Draw("hbar");
```

Hbar0,1,2,3,4



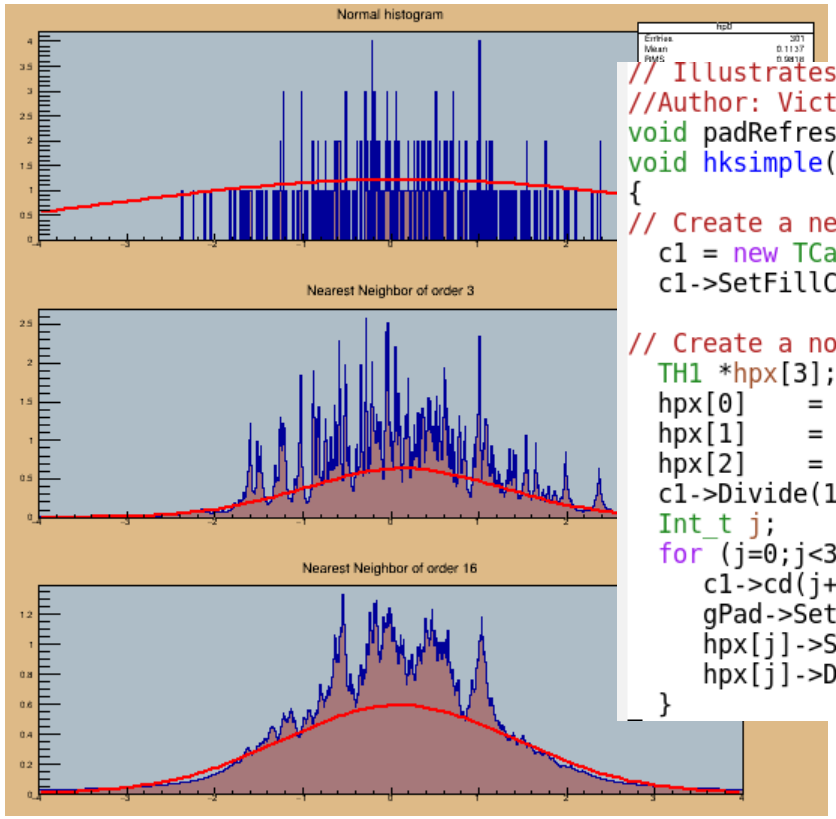


/tutorials/hist/hbars.C




```
//vertical bar chart
c1->cd(2); gPad->SetGrid(); gPad->SetFrameFillColor(33);
T->Draw("Division>>hDiv","","goff");
TH1F *hDiv   = (TH1F*)gDirectory->Get("hDiv");
hDiv->SetStats(0);
TH1F *hDivFR = (TH1F*)hDiv->Clone("hDivFR");
T->Draw("Division>>hDivFR","Nation==\"FR\"","goff");
hDiv->SetBarWidth(0.45);
hDiv->SetBarOffset(0.1);
hDiv->SetFillColor(49);
TH1 *h1 = hDiv->DrawCopy("bar2");
hDivFR->SetBarWidth(0.4);
hDivFR->SetBarOffset(0.55);
hDivFR->SetFillColor(50);
TH1 *h2 = hDivFR->DrawCopy("bar2,same");

TLegend *legend = new TLegend(0.55,0.65,0.76,0.82);
legend->AddEntry(h1,"All nations","f");
legend->AddEntry(h2,"French only","f");
legend->Draw();
```

/tutorials/hist/hksimple.C

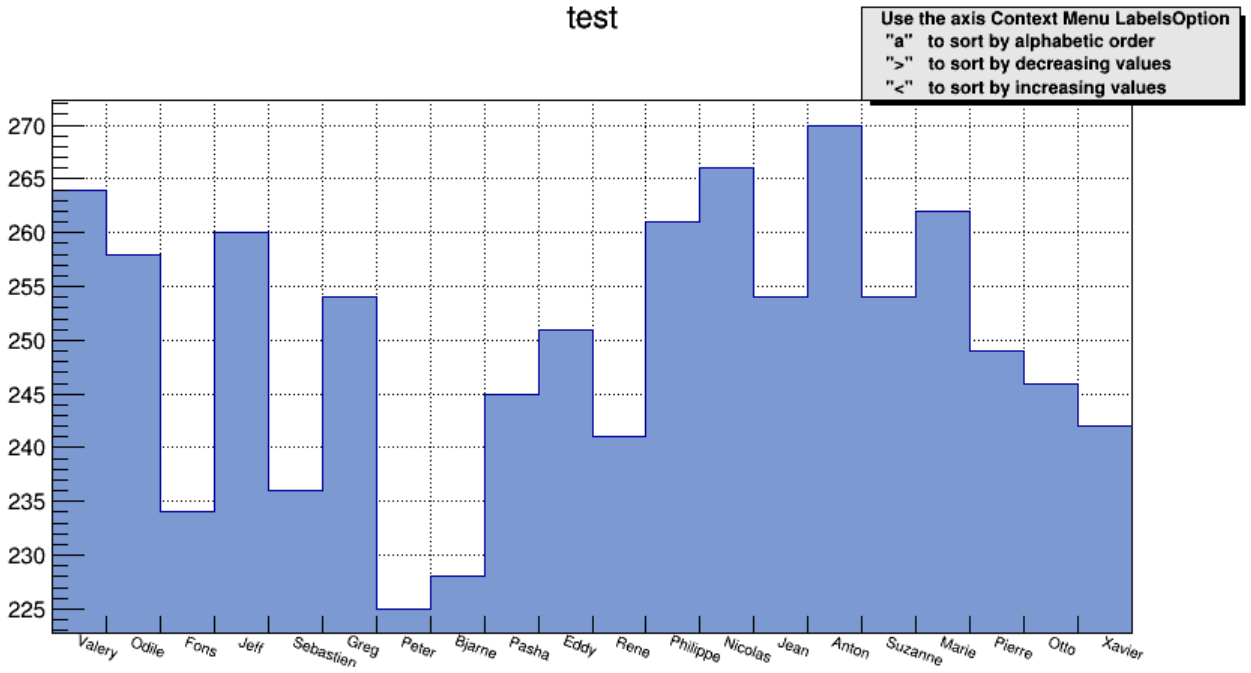



```
// Illustrates the advantages of a TH1K histogram
//Author: Victor Perevovchikov
void padRefresh(TPad *pad,int flag=0);
void hksimple()
{
// Create a new canvas.
  c1 = new TCanvas("c1","Dynamic Filling Example",200,10,600,900);
  c1->SetFillColor(42);

// Create a normal histogram and two TH1K histograms
  TH1 *hpx[3];
  hpx[0]    = new TH1F("hp0","Normal histogram",1000,-4,4);
  hpx[1]    = new TH1K("hk1","Nearest Neighbor of order 3",1000,-4,4);
  hpx[2]    = new TH1K("hk2","Nearest Neighbor of order 16",1000,-4,4,16);
  c1->Divide(1,3);
  Int_t j;
  for (j=0;j<3;j++) {
     c1->cd(j+1);
     gPad->SetFrameFillColor(33);
     hpx[j]->SetFillColor(48);
     hpx[j]->Draw();
  }
```

```
// Fill histograms randomly
   gRandom->SetSeed();
   Float_t px, py, pz;
   const Int_t kUPDATE = 10;
   for (Int_t i = 0; i <= 300; i++) {
      gRandom->Rannor(px,py);
      for (j=0;j<3;j++) {hpx[j]->Fill(px);}
      if (i && (i%kUPDATE) == 0) {
            padRefresh(c1);
      }
   }

   for (j=0;j<3;j++) hpx[j]->Fit("gaus");
   padRefresh(c1);
}
```

```
void padRefresh(TPad *pad,int flag)
{
  if (!pad) return;
  pad->Modified();
  pad->Update();
  TList *tl = pad->GetListOfPrimitives();
  if (!tl) return;
  TListIter next(tl);
  TObject *to;
  while ((to=next())) {
    if (to->InheritsFrom(TPad::Class())) padRefresh((TPad*)to,1);}
  if (flag) return;
  gSystem->ProcessEvents();
}
```

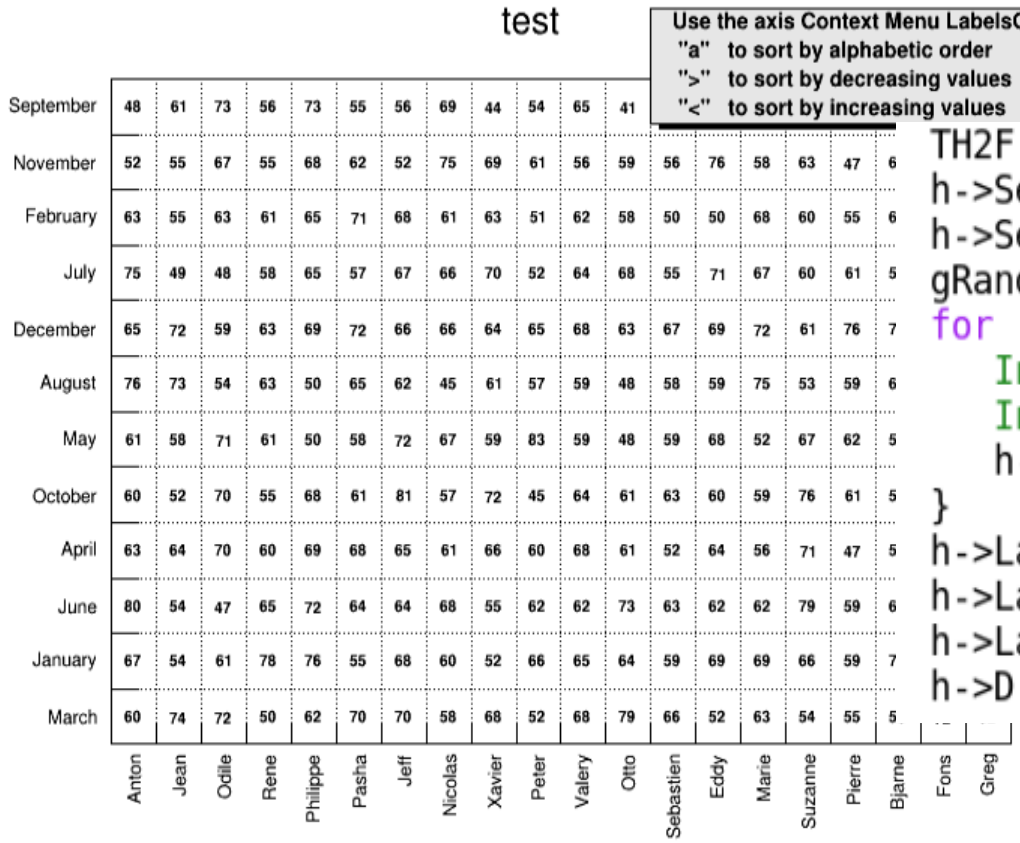
/tutorials/hist/hlabels1.C



siguang@pku.edu.cn

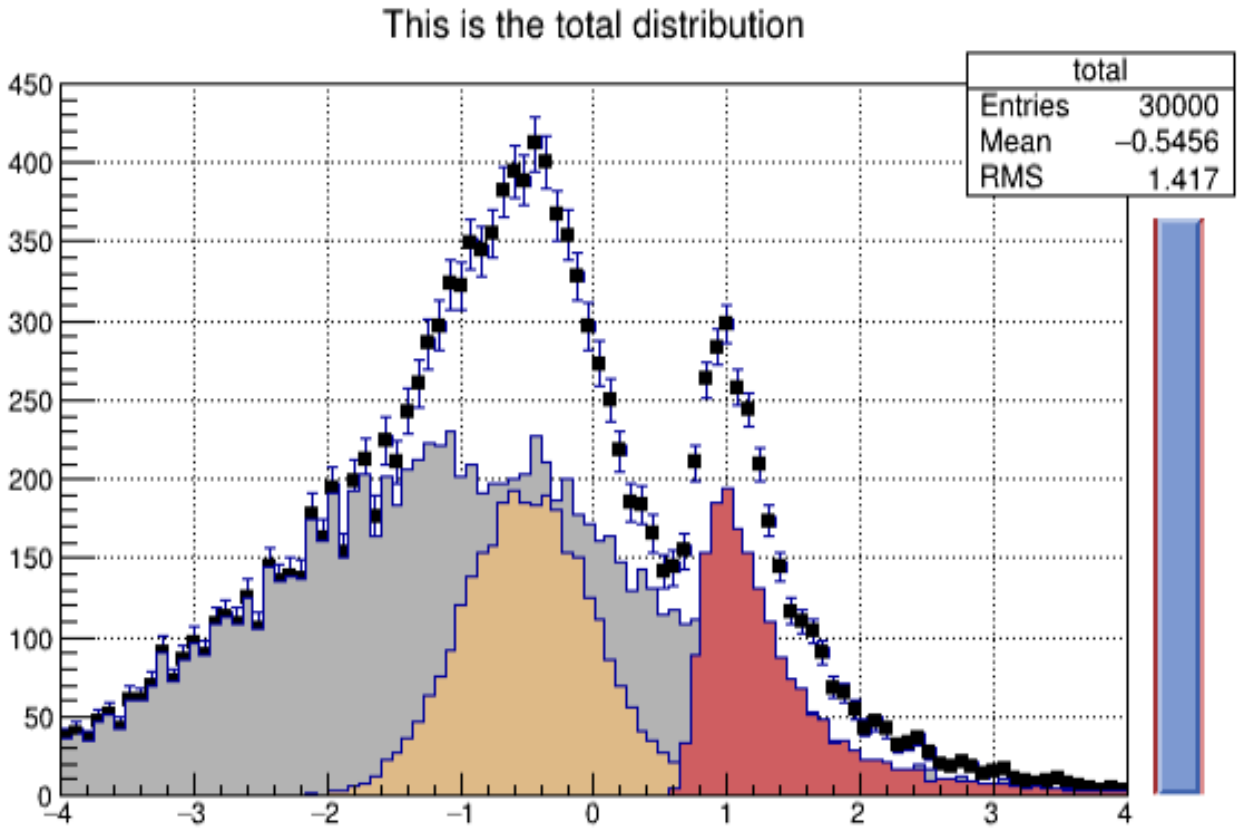
```
void hlabels1()
{
   const Int_t nx = 20;
   char *people[nx] = {"Jean","Pierre","Marie","Odile","Sebastien",
      "Fons","Rene","Nicolas","Xavier","Greg","Bjarne","Anton","Otto",
      "Eddy","Peter","Pasha","Philippe","Suzanne","Jeff","Valery"};
   TCanvas *c1 = new TCanvas("c1","demo bin labels",10,10,900,500);
   c1->SetGrid();
   c1->SetTopMargin(0.15);
   TH1F *h = new TH1F("h","test",3,0,3);
   h->SetStats(0);
   h->SetFillColor(38);
   h->SetBit(TH1::kCanRebin);
   for (Int_t i=0;i<5000;i++) {
      Int_t r = gRandom->Rndm()*20;
      h->Fill(people[r],1);
   }
   h->LabelsDeflate();
   h->Draw();
   ..................
```

/tutorials/hist/hlabels2.C




```
TH2F *h = new TH2F("h","test",3,0,3,2,0,2);
h->SetBit(TH1::kCanRebin);
h->SetStats(0);
gRandom->SetSeed();
for (Int_t i=0;i<15000;i++) {
   Int_t rx = gRandom->Rndm()*nx;
   Int_t ry = gRandom->Rndm()*ny;
   h->Fill(people[ry],month[rx],1);
}
h->LabelsDeflate("X");
h->LabelsDeflate("Y");
h->LabelsOption("v");
h->Draw("text");
```

/tutorials/hist/hsum.C

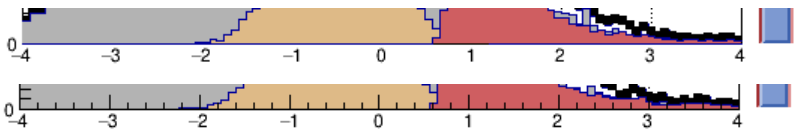


siguang@pku.edu.cn

```
  gBenchmark->Start("hsum");

// Create some histograms.
  total  = new TH1F("total","This is the total distribution",100,-4,4);
  main   = new TH1F("main","Main contributor",100,-4,4);
  s1     = new TH1F("s1","This is the first signal",100,-4,4);
  s2     = new TH1F("s2","This is the second signal",100,-4,4);
  total->Sumw2();  // store the sum of squares of weights
  total->SetMarkerStyle(21);
  total->SetMarkerSize(0.7);
  main->SetFillColor(16);
  s1->SetFillColor(42);
  s2->SetFillColor(46);
  TSlider *slider = 0;

// Fill histograms randomly
  gRandom->SetSeed();
  const Int_t kUPDATE = 500;
  Float_t xs1, xs2, xmain;
  for ( Int_t i=0; i<10000; i++) {
     xmain = gRandom->Gaus(-1,1.5);
```

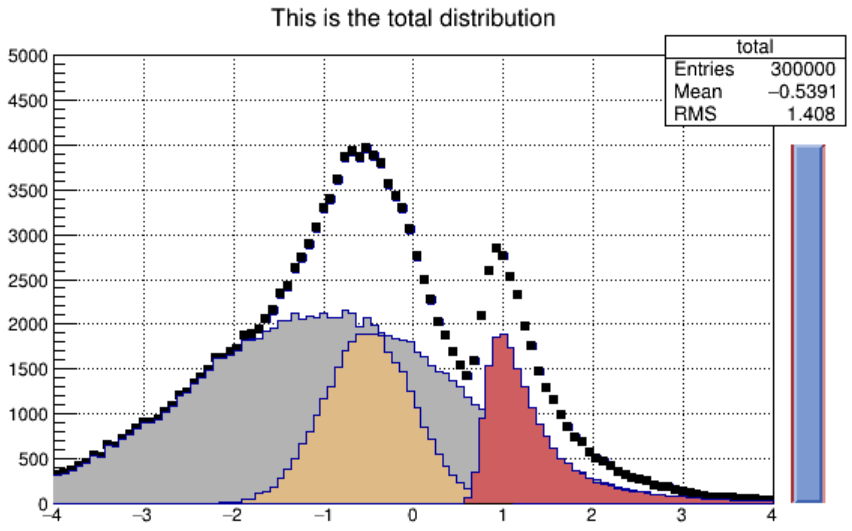
```
xs1   = gRandom->Gaus(-0.5,0.5);
xs2   = gRandom->Landau(1,0.15);
main->Fill(xmain);
s1->Fill(xs1,0.3);
s2->Fill(xs2,0.2);
total->Fill(xmain);
total->Fill(xs1,0.3);
total->Fill(xs2,0.2);
if (i && (i%kUPDATE) == 0) {
   if (i == kUPDATE) {
      total->Draw("e1p");
      main->Draw("same");
      s1->Draw("same");
      s2->Draw("same");
      c1->Update();
      slider = new TSlider("slider",
         "test",4.2,0,4.6,total->GetMaximum(),38);
      slider->SetFillColor(46);
   }
   if (slider) slider->SetRange(0,Float_t(i)/10000.);
```

```
        c1->Modified();
        c1->Update();
      }
    }
    slider->SetRange(0,1);
    total->Draw("sameaxis"); // to redraw axis hidden by the fill area
    c1->Modified();
    gBenchmark->Show("hsum");
}
```



```
root [0]
Processing hsum.C...
hsum      : Real Time =   1.42 seconds Cpu Time =   0.68 seconds
```

/tutorials/hist/hsumTimer.C

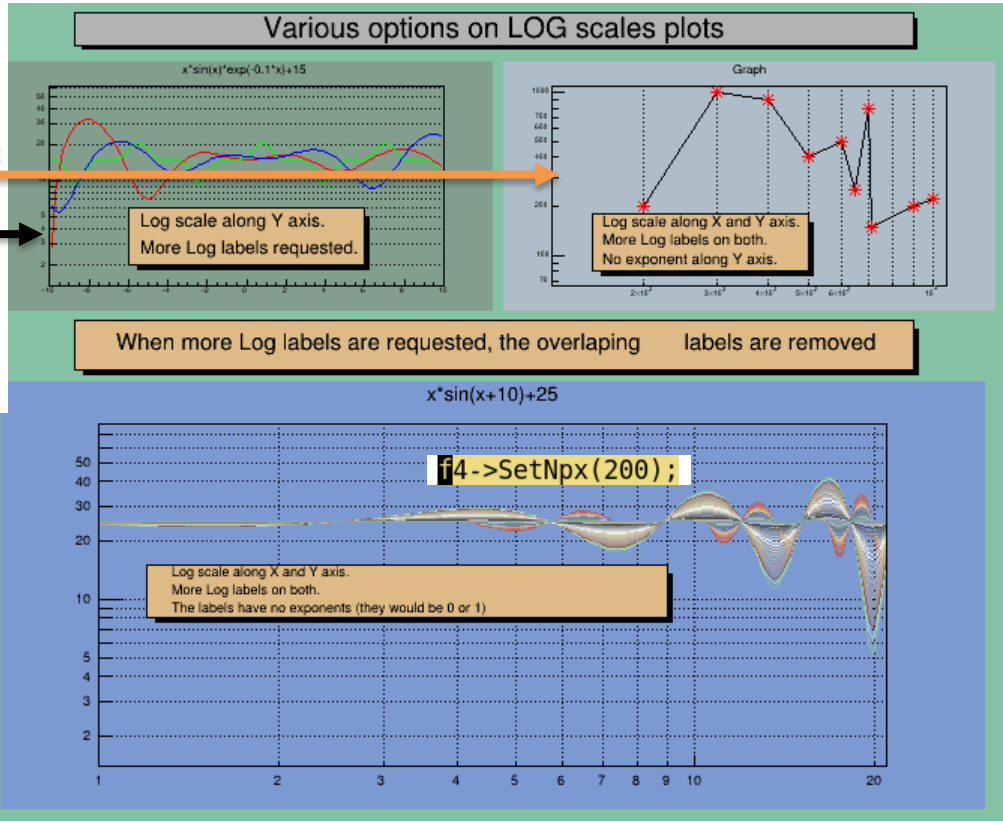



```
// Create a TTimer (hsumUpdate called every 300 msec)
 TTimer timer("hsumUpdate()",300);
 timer.TurnOn();

// Fill histograms randomly
 Float_t xs1, xs2, xmain;
 gRandom->SetSeed();
 for (Int_t i=0; i<nfill; i++) {
    ratio = Float_t(i)/Float_t(nfill);
    if (gSystem->ProcessEvents()) break;
    xmain = gRandom->Gaus(-1,1.5);
    xs1   = gRandom->Gaus(-0.5,0.5);
    xs2   = gRandom->Landau(1,0.15);
    main->Fill(xmain);
    s1->Fill(xs1,0.3);
    s2->Fill(xs2,0.2);
    total->Fill(xmain);
    ........
  }
  timer.TurnOff();
  hsumUpdate();
}

void hsumUpdate()
{
// called when Timer times out
    if (slider) slider->SetRange(0,ratio);
    c1->Modified();
    c1->Update();
}
```

/tutorials/hist/logscales.C

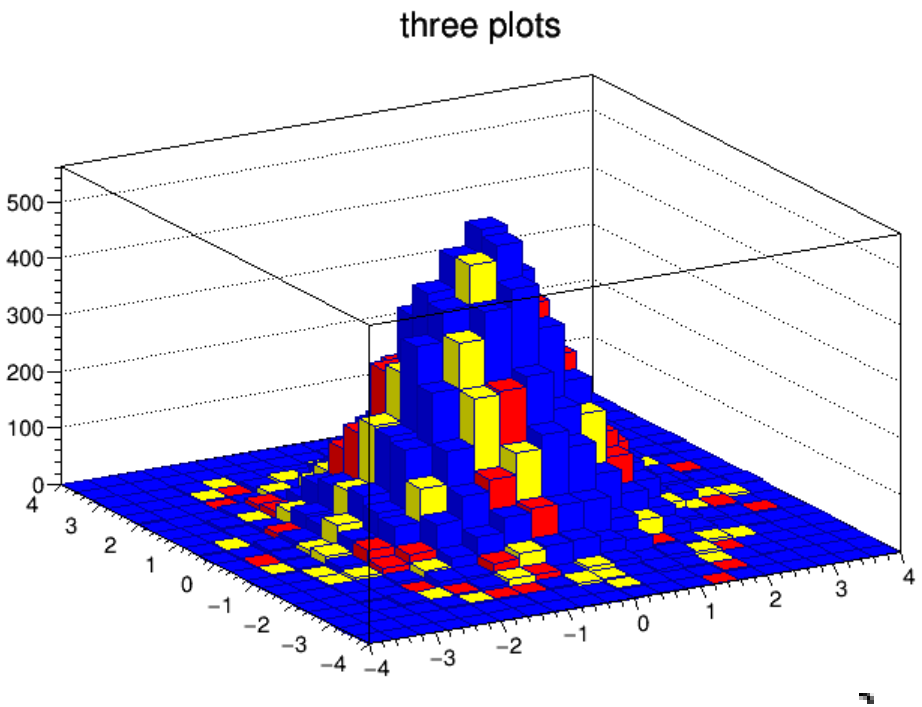
```
pad1->cd();
pad1->SetLogy();
pad1->SetGridy();
TF1 *f1 = new TF1("f1","x*sin(x)*exp(-0.1*x)+15",-10.,10.);
TF1 *f2 = new TF1("f2","(sin(x)+cos(x))**5+15",-10.,10.);
TF1 *f3 = new TF1("f3","(sin(x)/(x)-x*cos(x))+15",-10.,10.);
f1->SetLineWidth(1); f1->SetLineColor(2);
f2->SetLineWidth(1); f2->SetLineColor(3);
f3->SetLineWidth(1); f3->SetLineColor(4);
//f1->SetTitle("");
f1->Draw();
f2->Draw("same");
f3->Draw("same");
f1->GetYaxis()->SetMoreLogLabels();
```

```
g_2->GetYaxis()->SetMoreLogLabels();
g_2->GetYaxis()->SetNoExponent();
pad2->SetLogy();
g_2->GetXaxis()->SetMoreLogLabels();
```



siguang@pku.edu.cn

hist/multicolor.C

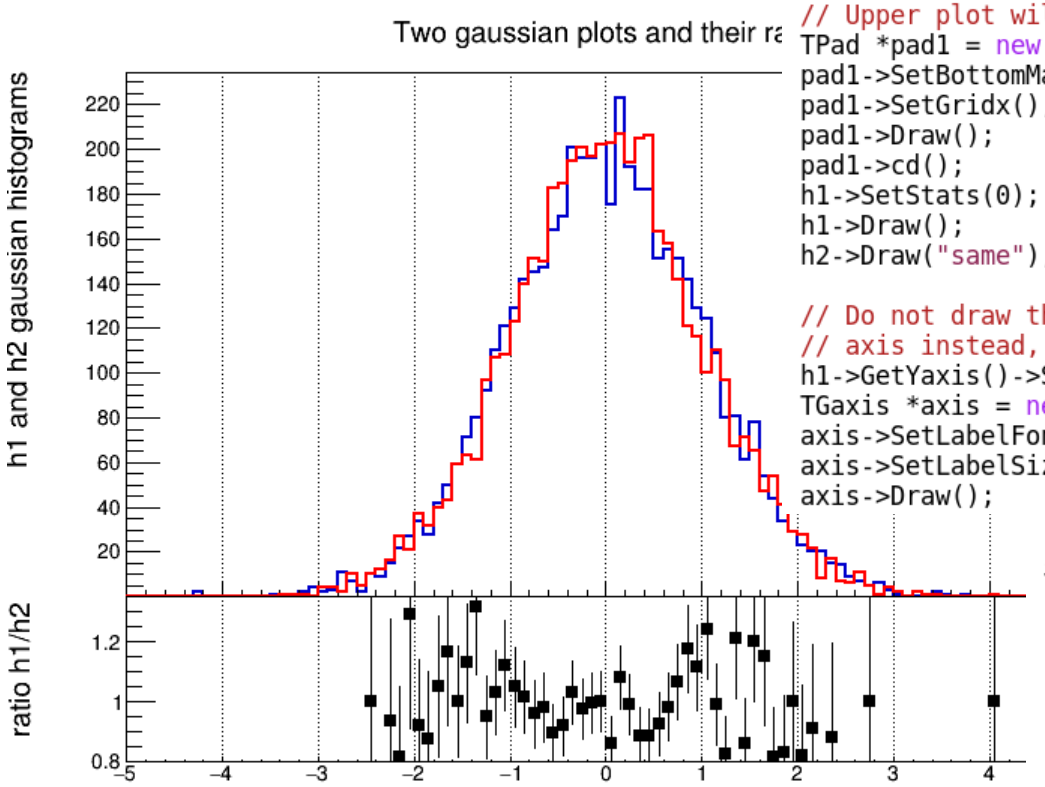



```
TH2F *h1 = new TH2F("h1","h1",nbins,-4,4,nbins,-4,4);
h1->SetFillColor(kBlue);
TH2F *h2 = new TH2F("h2","h2",nbins,-4,4,nbins,-4,4);
h2->SetFillColor(kRed);
TH2F *h3 = new TH2F("h3","h3",nbins,-4,4,nbins,-4,4);
h3->SetFillColor(kYellow);
THStack *hs = new THStack("hs","three plots");
hs->Add(h1);
hs->Add(h2);
hs->Add(h3);
```

.....

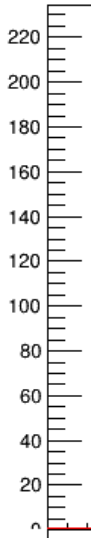
```
hs->Draw("lego1");
```

hist/ratioplot.C

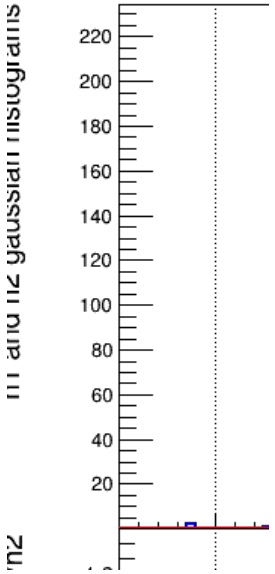



```
TCanvas *c = new TCanvas("c", "canvas", 800, 800);

// Upper plot will be in pad1
TPad *pad1 = new TPad("pad1", "pad1", 0, 0.3, 1, 1.0);
pad1->SetBottomMargin(0); // Upper and lower plot are joined
pad1->SetGridx();         // Vertical grid
pad1->Draw();             // Draw the upper pad: pad1
pad1->cd();               // pad1 becomes the current pad
h1->SetStats(0);          // No statistics on upper plot
h1->Draw();               // Draw h1
h2->Draw("same");         // Draw h2 on top of h1

// Do not draw the Y axis label on the upper plot and redraw a small
// axis instead, in order to avoid the first label (0) to be clipped.
h1->GetYaxis()->SetLabelSize(0.);
TGaxis *axis = new TGaxis( -5, 20, -5, 220, 20,220,510,"");
axis->SetLabelFont(43); // Absolute font size in pixel (precision 3)
axis->SetLabelSize(15);
axis->Draw();
```

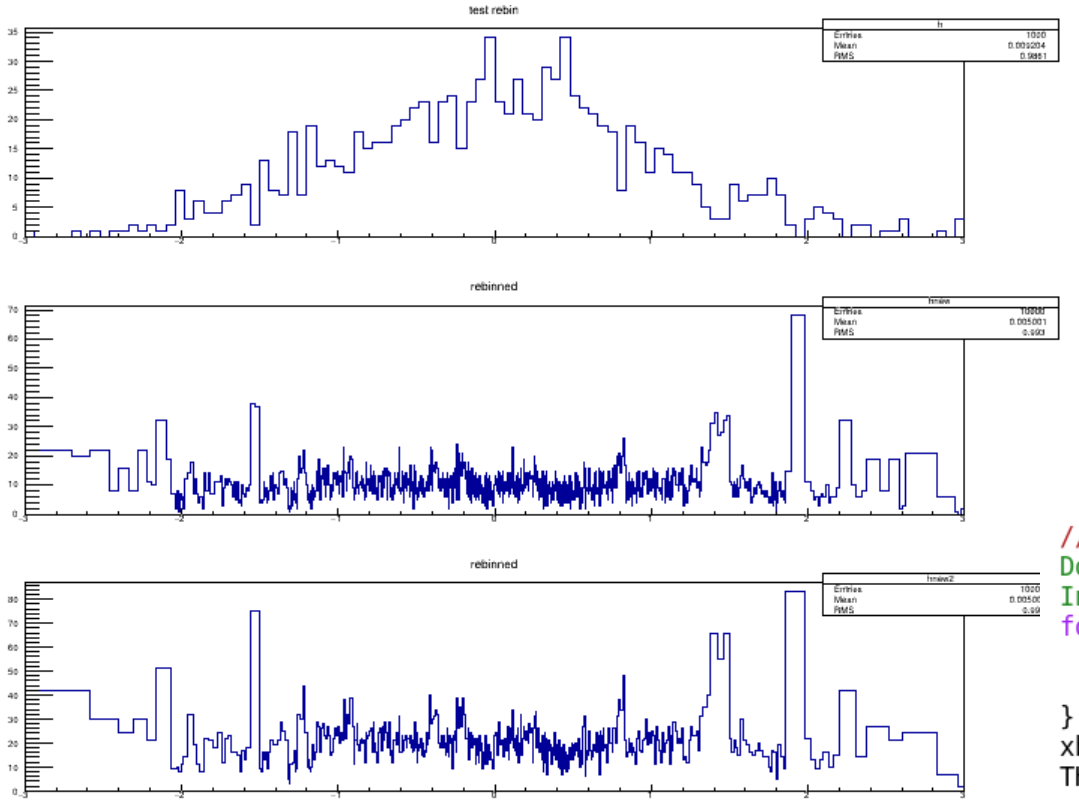
```
// Define the ratio plot
TH1F *h3 = (TH1F*)h1->Clone("h3");
h3->SetLineColor(kBlack);
h3->SetMinimum(0.8);  // Define Y ..
h3->SetMaximum(1.35); // .. range
h3->Sumw2();
h3->SetStats(0);      // No statistics on lower plot
h3->Divide(h2);
```




```
//  h1->GetYaxis()->SetLabelSize(0.);
h1->SetMinimum(0.001);
```



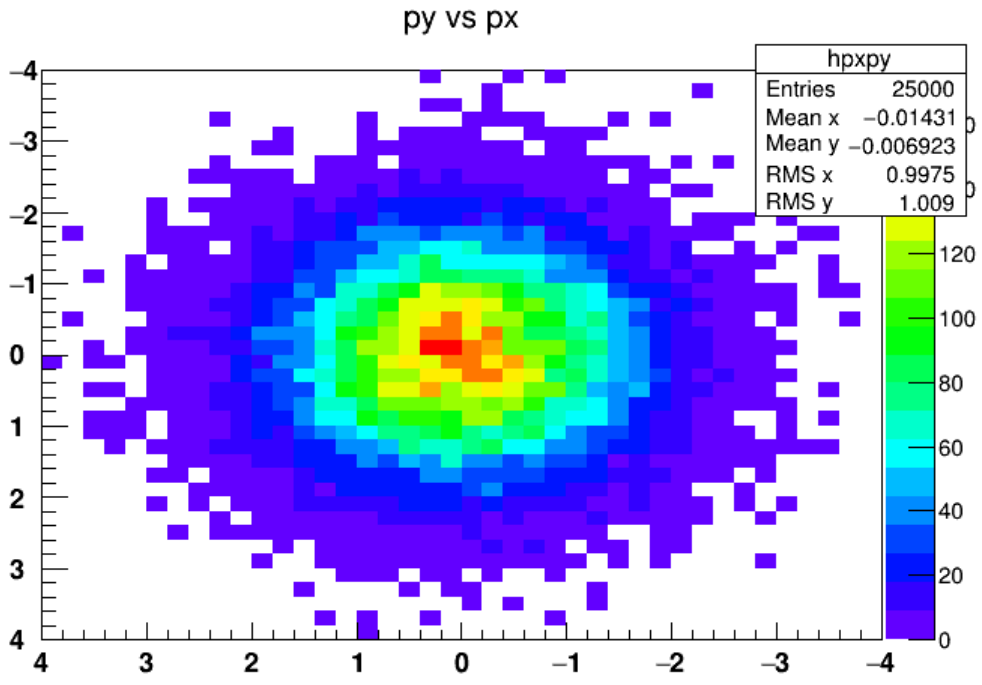
hist/rebin.C




```
//rebin hnew keeping only 50% of the bins
Double_t xbins2[501];
Int_t kk=0;
for (Int_t j=0;j<k;j+=2) {
   xbins2[kk] = xbins[j];
   kk++;
}
xbins2[kk] = xbins[k];
TH1F *hnew2 = (TH1F*)hnew->Rebin(kk,"hnew2",xbins2);
```

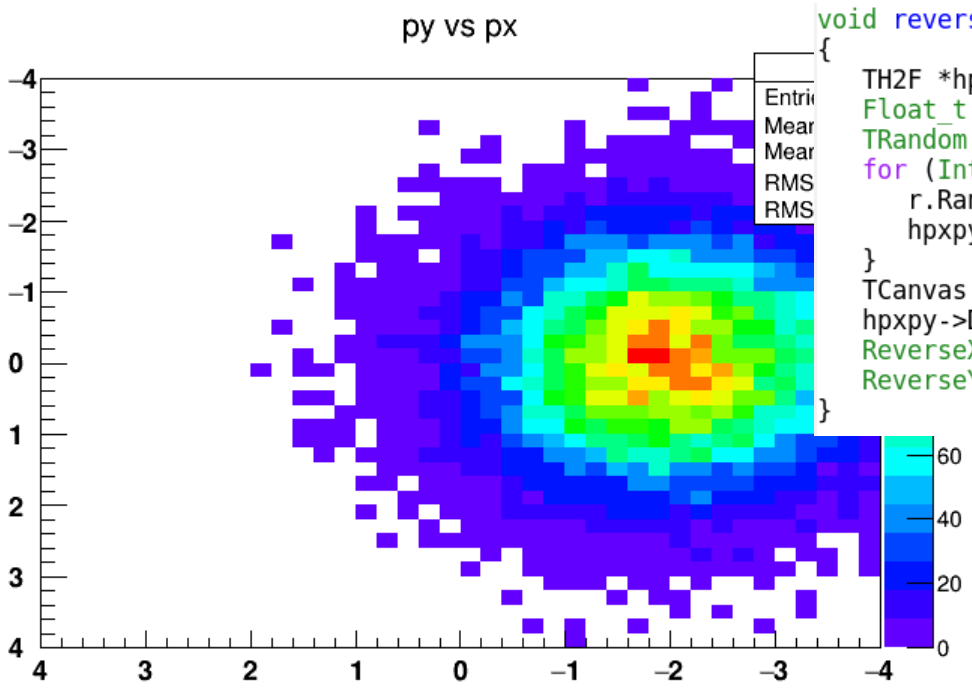

siguang@pku.edu.cn

hist/reverseaxis.C




```
void ReverseXAxis (TH1 *h)
{
   // Remove the current axis
   h->GetXaxis()->SetLabelOffset(999);
   h->GetXaxis()->SetTickLength(0);

   // Redraw the new axis
   gPad->Update();
   TGaxis *newaxis = new TGaxis(gPad->GetUxmax(),
                                gPad->GetUymin(),
                                gPad->GetUxmin(),
                                gPad->GetUymin(),
                                h->GetXaxis()->GetXmin(),
                                h->GetXaxis()->GetXmax(),
                                510,"-");
   newaxis->SetLabelOffset(-0.03);
   newaxis->Draw();
}
```

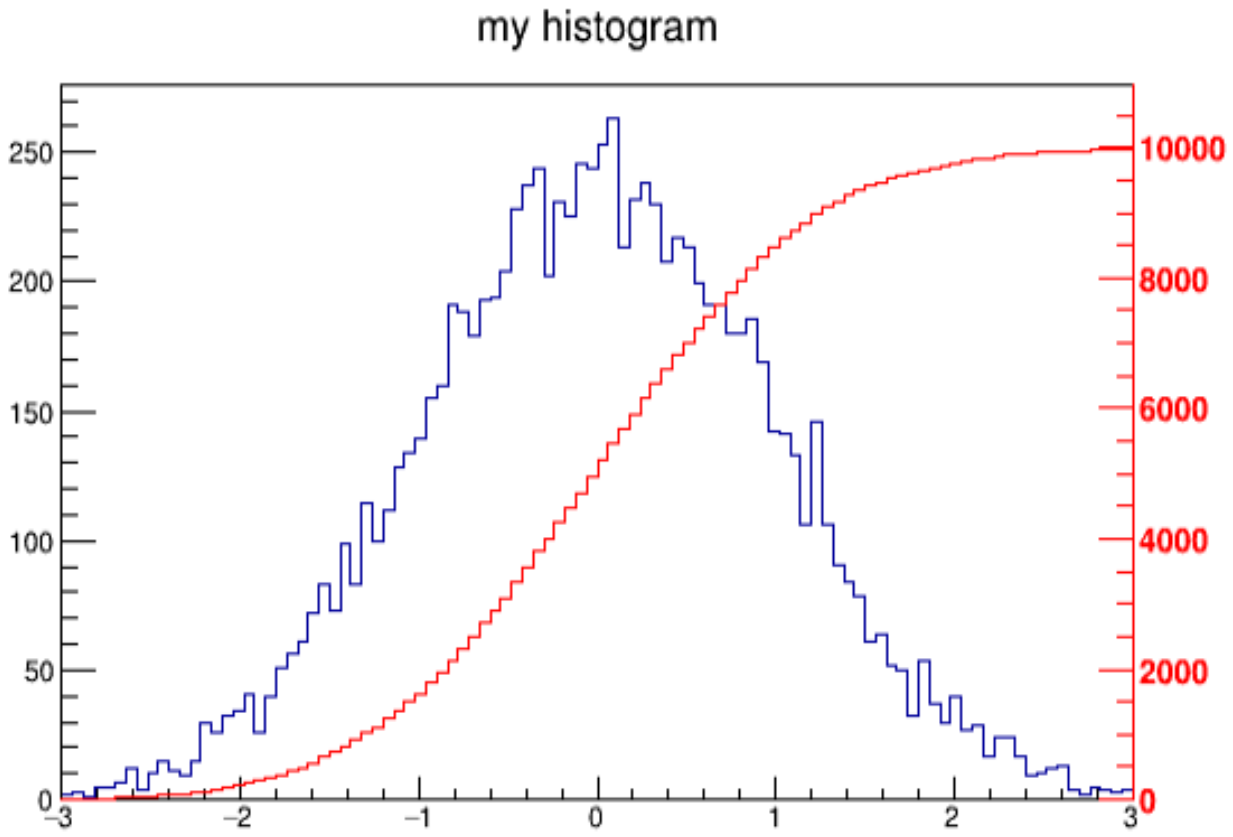
More test hist/reverseaxis.C




```
void reverseaxis ()
{
   TH2F *hpxpy  = new TH2F("hpxpy","py vs px",40,-4,4,40,-4,4);
   Float_t px, py;
   TRandom r;
   for (Int_t i = 0; i < 25000; i++) {
      r.Rannor(px,py);
      hpxpy->Fill(px+2,py);
   }
   TCanvas *c1 = new TCanvas(
   hpxpy->Draw("colz");
   ReverseXAxis(hpxpy);
   ReverseYAxis(hpxpy);
}
```

+2 to test the new X-axis has nothing to do with data

hist/twoscales.C

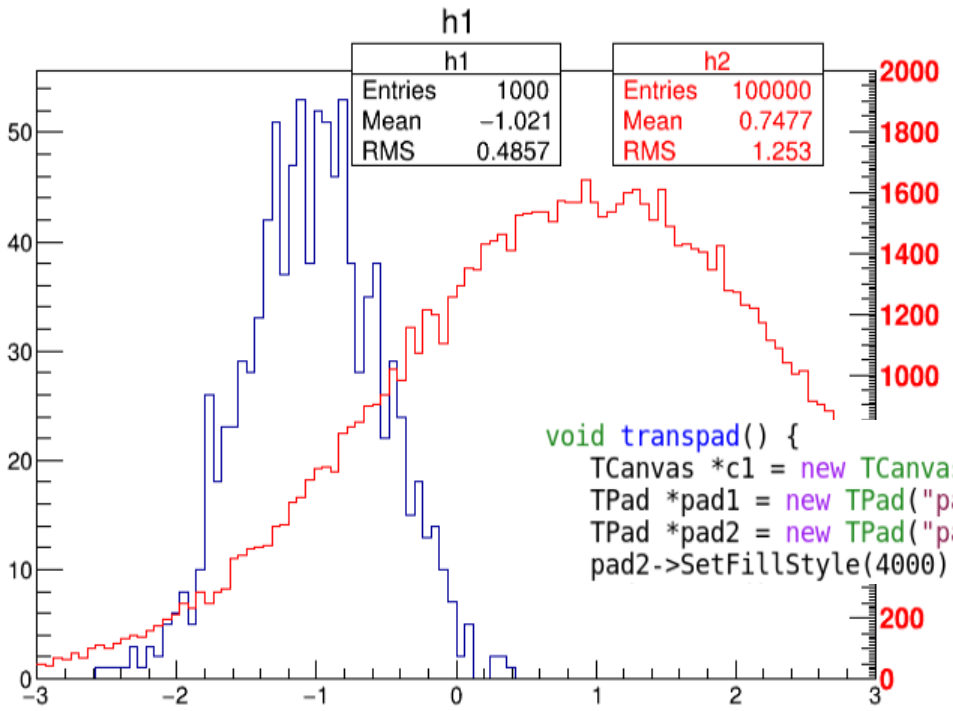


siguang@pku.edu.cn

```
//create/fill draw h1
gStyle->SetOptStat(kFALSE);
TH1F *h1 = new TH1F("h1","my histogram",100,-3,3);
Int_t i;
for (i=0;i<10000;i++) h1->Fill(gRandom->Gaus(0,1));
h1->Draw();
c1->Update();

//create hint1 filled with the bins integral of h1
TH1F *hint1 = new TH1F("hint1","h1 bins integral",100,-3,3);
Float_t sum = 0;
for (i=1;i<=100;i++) {
   sum += h1->GetBinContent(i);
   hint1->SetBinContent(i,sum);
}

//scale hint1 to the pad coordinates
Float_t rightmax = 1.1*hint1->GetMaximum();
Float_t scale = gPad->GetUymax()/rightmax;
hint1->SetLineColor(kRed);
hint1->Scale(scale);
hint1->Draw("same");
```

```
//draw an axis on the right side
TGaxis *axis = new TGaxis(gPad->GetUxmax(),gPad->GetUymin(),
      gPad->GetUxmax(), gPad->GetUymax(),0,rightmax,510,"+L");
axis->SetLineColor(kRed);
axis->SetLabelColor(kRed);
axis->Draw();
```

hist/transpad.C




```
void transpad() {
   TCanvas *c1 = new TCanvas("c1","transparent pad",200,10,700,500);
   TPad *pad1 = new TPad("pad1","",0,0,1,1);
   TPad *pad2 = new TPad("pad2","",0,0,1,1);
   pad2->SetFillStyle(4000); //will be transparent
```

SetFillStyle

Conventions for fill styles:

- 0 : hollow
- 1001 : Solid
- 2001 : hatch style
- 3000+pattern_number (see next page)

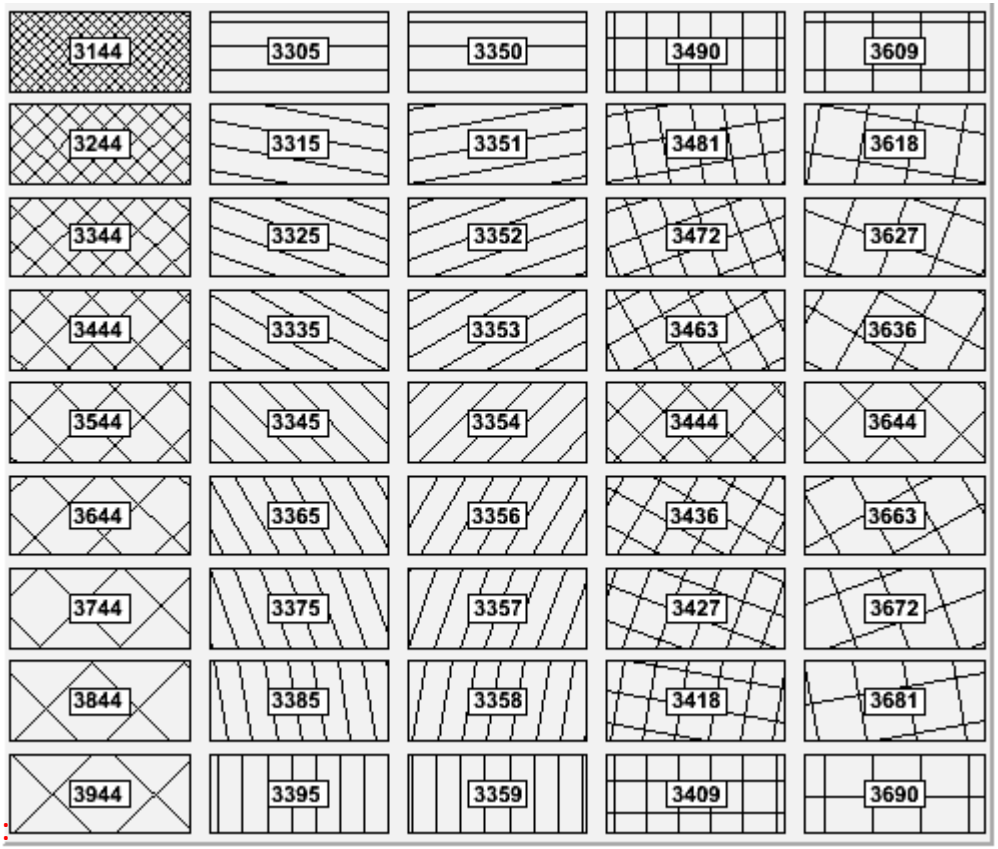
For TPad only:

- 4000 :the window is transparent.
  - 4000 to 4100 the window is 100% transparent to 100% opaque.

The pad transparency is visible in binary outputs files like gif, jpg, png etc .. but not in vector graphics output files like PS, PDF and SVG.

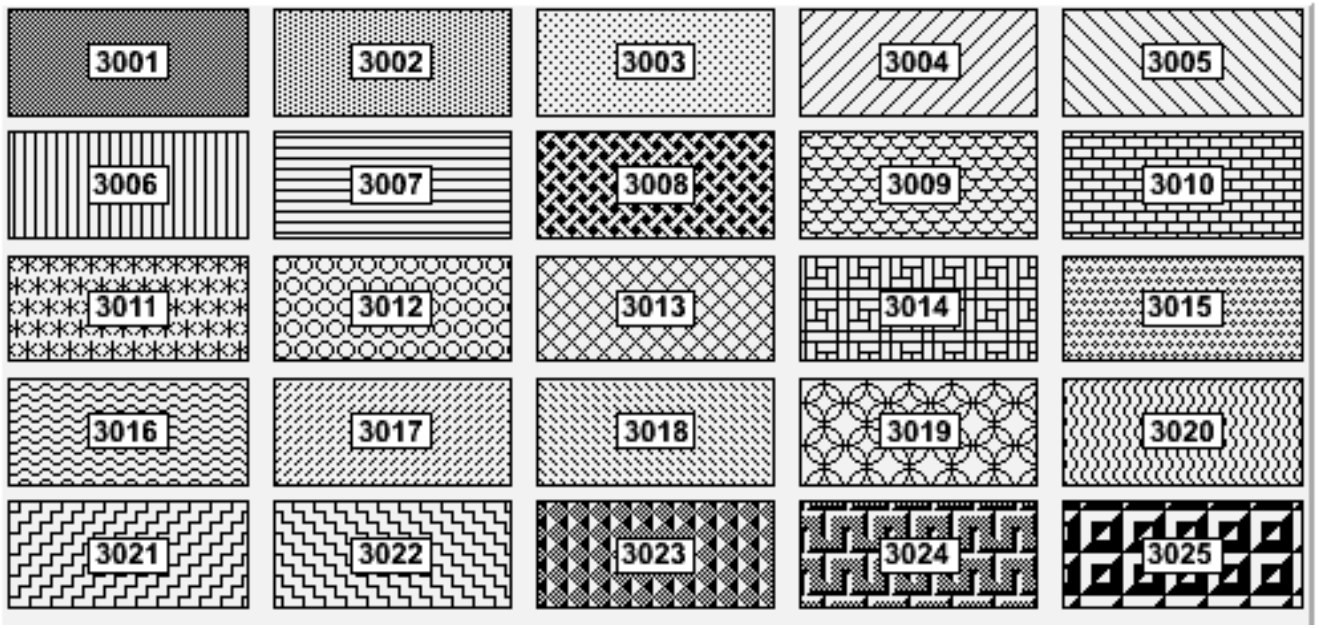
pattern_number = ijk (FillStyle = 3ijk)

- i (1-9) : specify the space between each hatch
  - 1 = 1/2mm 9 = 6mm
- j (0-9) : specify angle between 0 and 90 de
  - 0 = 0
  - 1 = 10
  - 2 = 20
  - 3 = 30
  - 4 = 45
  - 5 = Not drawn
  - 6 = 60
  - 7 = 70
  - 8 = 80
  - 9 = 90
- k (0-9) : specify angle between 90 and 18(
  - 0 = 180
  - 1 = 170
  - 2 = 160
  - 3 = 150
  - 4 = 135
  - 5 = Not drawn
  - 6 = 120
  - 7 = 110
  - 8 = 100
  - 9 = 90

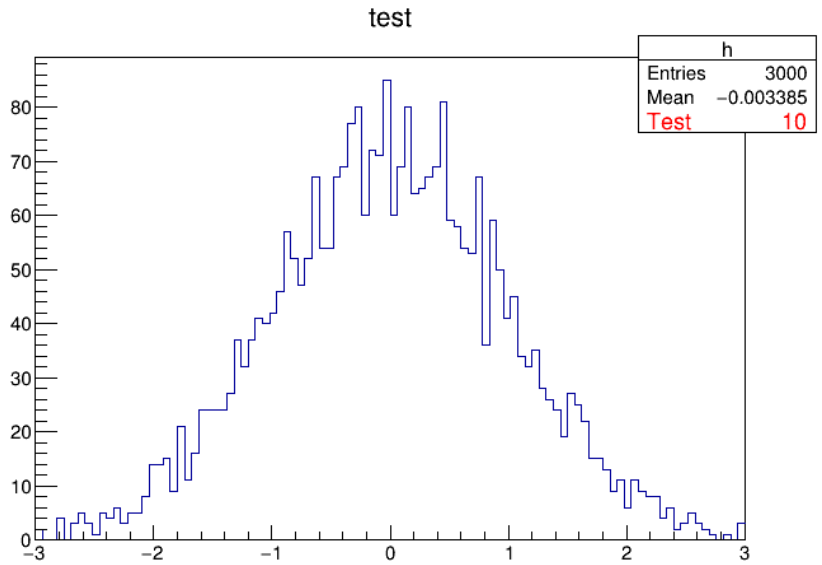


Note:
gStyle->SetHatchesSpacing() to define the spacing between hatches.
gStyle->SetHatchesLineWidth() to define the hatches line width.

25 Fixed Styles



hist/statsEditing.C

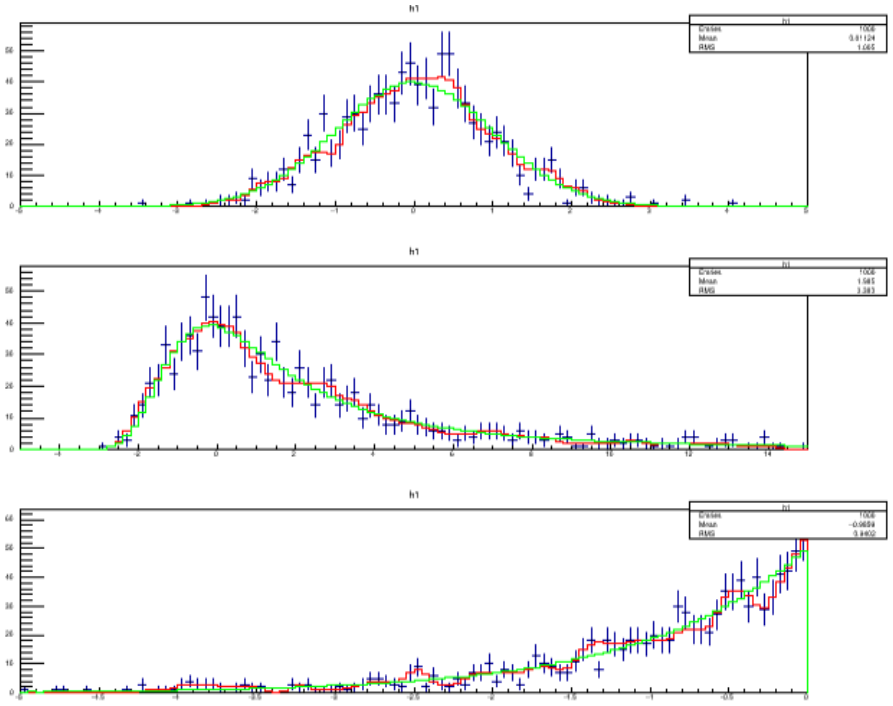



```
// Create and plot a test histogram with stats
TCanvas *se = new TCanvas;
TH1F *h = new TH1F("h","test",100,-3,3);
h->FillRandom("gaus",3000);
gStyle->SetOptStat();
h->Draw();
se->Update();

// Retrieve the stat box
TPaveStats *ps = (TPaveStats*)se->GetPrimitive("stats");
ps->SetName("mystats");
TList *list = ps->GetListOfLines();

// Remove the RMS line
TText *tconst = ps->GetLineWith("RMS");
list->Remove(tconst);

// Add a new line in the stat box.
// Note that "=" is a control character
TLatex *myt = new TLatex(0,0,"Test = 10");
myt ->SetTextFont(42);
myt ->SetTextSize(0.04);
myt ->SetTextColor(kRed);
list->Add(myt);

// the following line is needed to avoid that the automatic redrawing of stats
h->SetStats(0);

se->Modified();
return se;
```

hist/testSmooth.C

```
void smooth_hist(const char * fname, double xmin, double xmax, int n1, int n2) {

   std::cout << "somoothing a " << fname << " histogram" << std::endl;

   TH1D * h1 = new TH1D("h1","h1",100,xmin,xmax);
   TH1D * h2 = new TH1D("h2","h2",100,xmin,xmax);
   h1->FillRandom(fname,n1);

   TH1D * h1_s = new TH1D(*h1);
   h1_s->SetName("h1_s");
   h1_s->Smooth();

   h2->FillRandom(fname,n2);

   double p1 = h1->Chi2Test(h2,"");
   double p2 = h1_s->Chi2Test(h2,"UU");
   if (p2 < p1) Error("testSmooth","TH1::Smooth is
ed");
```

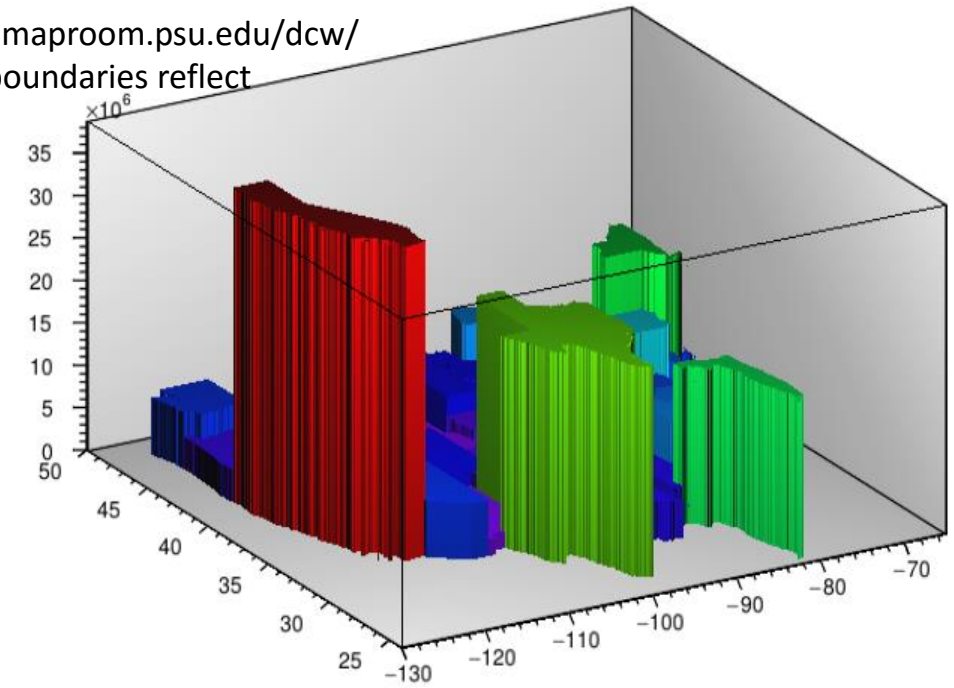
- option:
  - "UU" = experiment experiment comparison (unweighted-unweighted)
  - "UW" = experiment MC comparison (unweighted-weighted). Note that the first histogram should be unweighted
  - "WW" = MC MC comparison (weighted-weighted)



/tutorials/hist/th2polyUSA.C

```
//This tutorial illustrates how to create an histogram with polygonal
//bins (TH2Poly), fill it and draw it using GL. The initial data are stored
//in TMultiGraphs. They represent the USA.
//
//The initial data have been downloaded from: http://www.maproom.psu.edu/dcw/
//This database was developed in 1991/1992 and national boundaries reflect
//political reality as of that time.
//
//Author: Olivier Couet
```

https://root.cern.ch/download/doc/ROOTUsersGuideHTML/ch03s15.html



TH2Poly

TH2Poly is a 2D Histogram class allowing to define polygonal bins of arbitary shape.
Each bin in the TH2Poly histogram is a TH2PolyBin object. TH2PolyBin is a very simple class containing the vertices and contents of the polygonal bin as well as several related functions.
Bins are defined using one of the AddBin() methods. The bin definition should be done before filling.

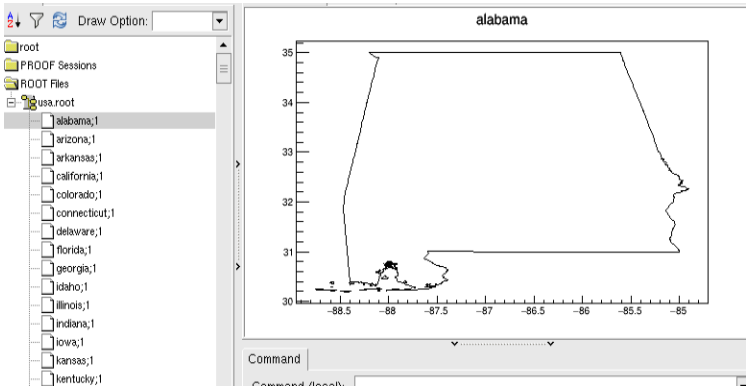
```
{
   TH2Poly *h2p = new TH2Poly();
   Double_t x1[] = {0, 5, 6};
   Double_t y1[] = {0, 0, 5};
   Double_t x2[] = {0, -1, -1, 0};
   Double_t y2[] = {0, 0, -1, 3};
   Double_t x3[] = {4, 3, 0, 1, 2.4};
   Double_t y3[] = {4, 3.7, 1, 3.7, 2.5};
   h2p->AddBin(3, x1, y1);
   h2p->AddBin(4, x2, y2);
   h2p->AddBin(5, x3, y3);
   h2p->Fill(0.1, 0.01, 3);
   h2p->Fill(-0.5, -0.5, 7);
   h2p->Fill(-0.7, -0.5, 1);
   h2p->Fill(1, 3, 1.5);
}
```

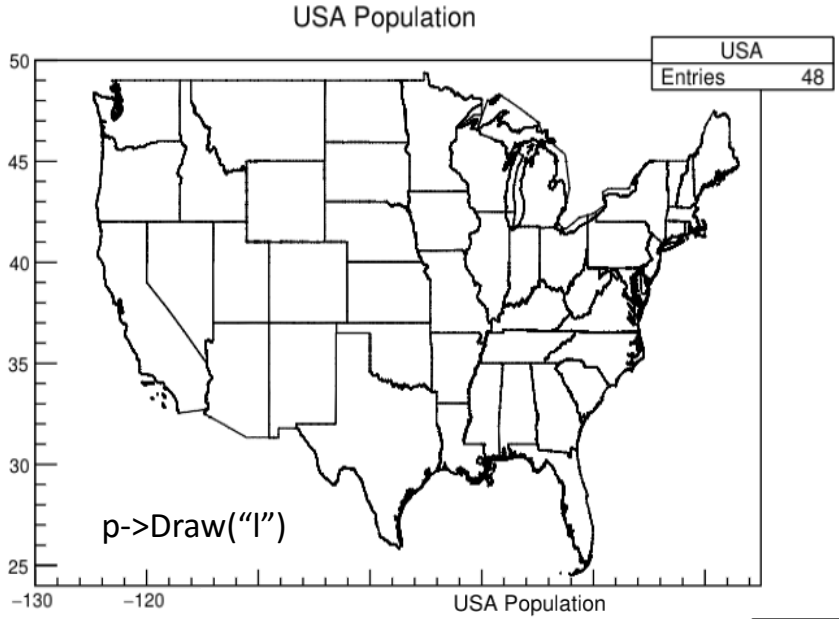
https://root.cern.ch/root/html534/TH2Poly.html

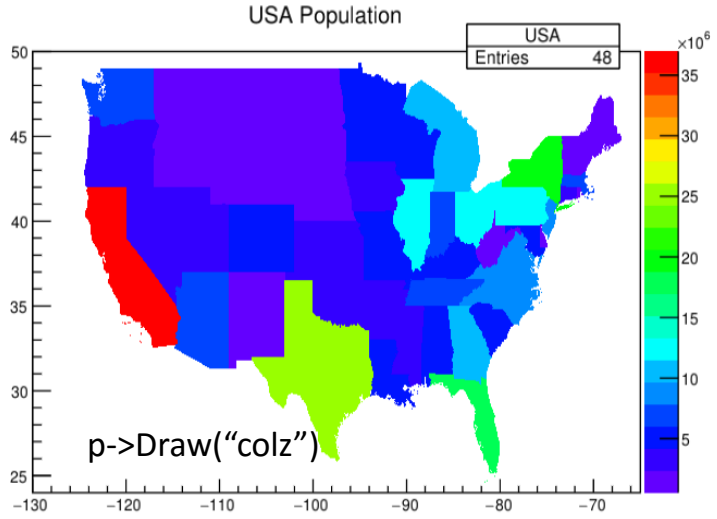
```
void th2polyUSA()
{
   Int_t i, bin;
   const Int_t nx = 48;
   char *states [nx] = {
      "alabama",      "arizona",        "arkansas",       "california",
      "colorado",     "connecticut",    "delaware",       "florida",
      "georgia",      "idaho",          "illinois",       "indiana",
      "iowa",         "kansas",         "kentucky",       "louisiana",
      "maine",        "maryland",       "massachusetts",  "michigan",
      "minnesota",    "mississippi",    "missouri",       "montana",
      "nebraska",     "nevada",         "new_hampshire",  "new_jersey",
      "new_mexico",   "new_york",       "north_carolina", "north_dakota",
      "ohio",         "oklahoma",       "oregon",         "pennsylvania",
      "rhode_island", "south_carolina", "south_dakota",   "tennessee",
      "texas",        "utah",           "vermont",        "virginia",
      "washington",   "west_virginia",  "wisconsin",      "wyoming"
   };
   Double_t pop[nx] = {
    4708708, 6595778,  2889450, 36961664, 5024748,  3518288,  885122, 18537969,
    9829211, 1545801, 12910409,  6423113, 3007856,  2818747, 4314113,  4492076,
    1318301, 5699478,  6593587,  9969727, 5266214,  2951996, 5987580,   974989,
    1796619, 2643085,  1324575,  8707739, 2009671, 19541453, 9380884,   646844,
   11542645, 3687050,  3825657, 12604767, 1053209,  4561242,  812383,  6296254,
   24782302, 2784572,   621760,  7882590, 6664195,  1819777, 5654774,   544270
   };
```

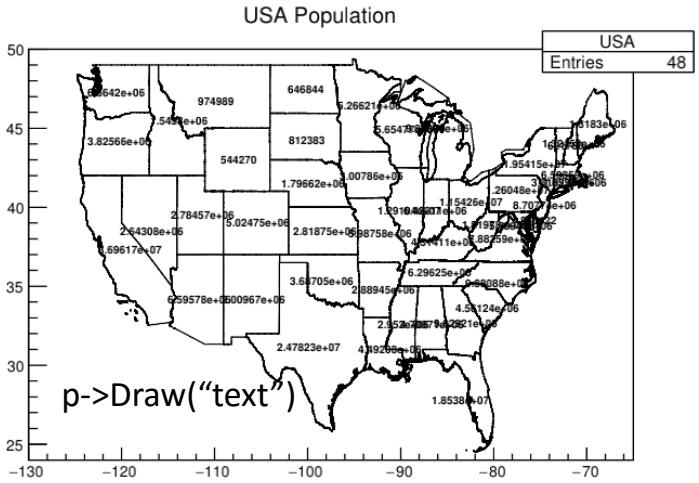
```
gStyle->SetCanvasPreferGL(true);
TCanvas *usa = new TCanvas("USA", "USA");
usa->ToggleEventStatus();
Double_t lon1 = -130;
Double_t lon2 = -65;
Double_t lat1 = 24;
Double_t lat2 = 50;
TH2Poly *p = new TH2Poly("USA","USA Population",lon1,lon2,lat1,lat2);

TFile *f;
f = TFile::Open("http://root.cern.ch/files/usa.root");
//use: curl -o usa.root  http://root.cern.ch/files/usa.root
//f = TFile::Open("usa.root");
if (!f) {
   printf("Cannot access usa.root. Is internet working ?\n");
   return;
}
```

```
   // Define the TH2Poly bins.
   TMultiGraph *mg;
   TKey *key;
   TIter nextkey(gDirectory->GetListOfKeys());
   while (key = (TKey*)nextkey()) {
      obj = key->ReadObj();
      if (obj->InheritsFrom("TMultiGraph")) {
         mg = (TMultiGraph*)obj;
         bin = p->AddBin(mg);
      }
   }

   // Fill TH2Poly.
   for (i=0; i<nx; i++) p->Fill(states[i], pop[i]);

   gStyle->SetOptStat(11);
   gStyle->SetPalette(1);
   p->Draw("legogl");
}
```


Draw Option:
root
PROOF Sessions
ROOT Files
usa.root
alabama;1
arizona;1
arkansas;1
california;1
colorado;1
connecticut;1
delaware;1
florida;1
georgia;1
idaho;1
illinois;1
indiana;1
iowa;1
kansas;1
kentucky;1
alabama
Command

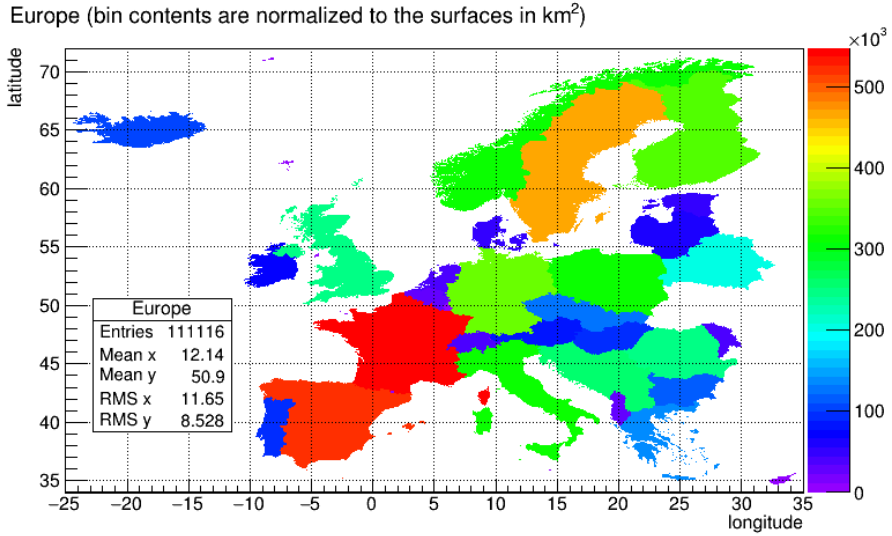

USA Population
USA
Entries 48
p->Draw("l")

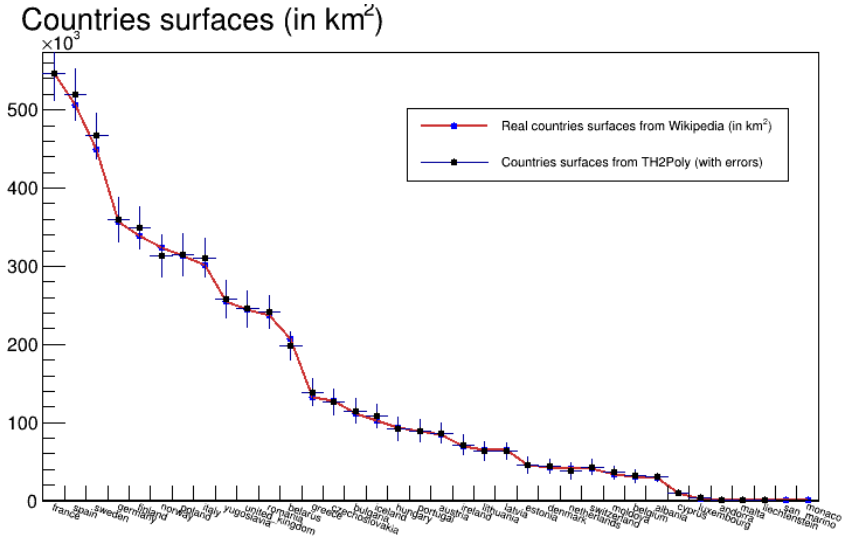

USA Population
USA
Entries 48
p->Draw("colz")


USA Population
USA
Entries 48
p->Draw("text")

/tutorials/hist/root th2polyEurope.C

```
//The script is shooting npoints (script argument) randomly over the Europe area.
//The number of points inside the countries should be proportional to the country surface
//The estimated surface is compared to the surfaces taken from wikipedia.
//Author: Olivier Couet
```

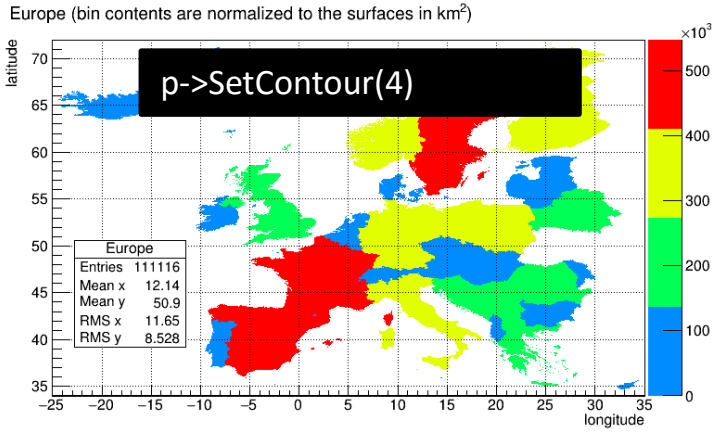




siguang@pku.edu.cn

```
char *countries[nx] = { "france",     "spain",  "sweden",  "germany",       "finland",
   "norway",     "poland", "italy",   "yugoslavia",    "united_kingdom",
   "romania",    "belarus","greece",  "czechoslovakia","bulgaria",
   "iceland",    "hungary","portugal","austria",       "ireland",
   "lithuania",  "latvia", "estonia", "denmark",       "netherlands",
   "switzerland","moldova","belgium", "albania",       "cyprus",
   "luxembourg", "andorra","malta",   "liechtenstein", "san_marino",
   "monaco" };
Float_t surfaces[nx] = { 547030,        505580,   449964,      357021,        338145,
   324220,        312685,   301230,      255438,        244820,
   237500,        207600,   131940,      127711,        110910,
   103000,         93030,    89242,       83870,         70280,
   65200,         64589,    45226,       43094,         41526,
   41290,         33843,    30528,       28748,          9250,
   2586,           468,      316,         160,            61,
   2};

TH1F *h = new TH1F("h","Countries surfaces (in km^{2})",3,0,3);
for (i=0; i<nx; i++) h->Fill(countries[i], surfaces[i]);
h->LabelsDeflate();
```

```
TFile::SetCacheFileDir(".");
TFile *f;
f = TFile::Open("http://root.cern.ch/files/europe.root","cacheread");
//or predownload curl -o europe.root http://root.cern.ch/files/europe.root
//f = TFile::Open("europe.root");
if (!f) {
   printf("Cannot access europe.root. Is internet working ?\n");
   return;
}

TH2Poly *p = new TH2Poly(
          "Europe",
          "Europe (bin contents are normalized to the surfaces in km^{2})",
          lon1,lon2,lat1,lat2);
p->GetXaxis()->SetNdivisions(520);
p->GetXaxis()->SetTitle("longitude");
p->GetYaxis()->SetTitle("latitude");

p->SetContour(100);
```



```
TMultiGraph *mg;
TKey *key;
TIter nextkey(gDirectory->GetListOfKeys());
while (key = (TKey*)nextkey()) {
   obj = key->ReadObj();
   if (obj->InheritsFrom("TMultiGraph")) {
      mg = (TMultiGraph*)obj;
      p->AddBin(mg);
   }
}

TRandom r;
Double_t longitude, latitude;
Double_t x, y, pi4 = TMath::Pi()/4, alpha = TMath::Pi()/360;

gBenchmark->Start("Partitioning");
p->ChangePartition(100, 100);
```

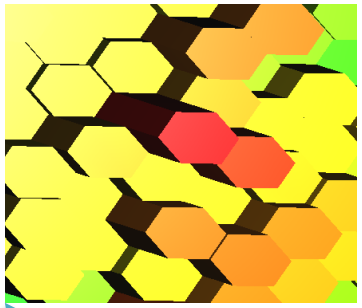
Changes the number of partition cells in the histogram.
Deletes the old partition and constructs a new one.

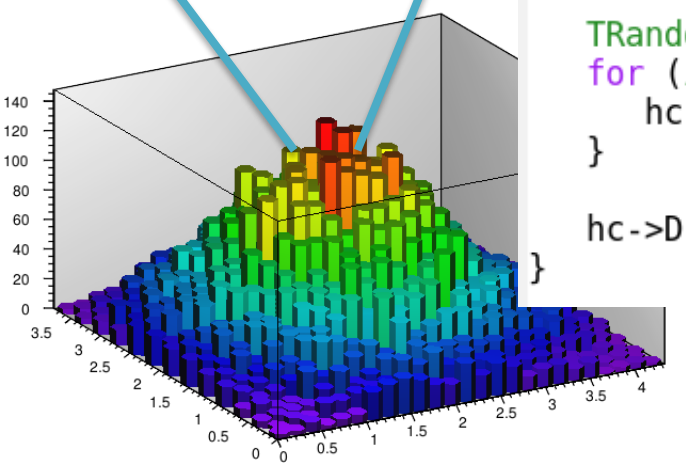
```
// Fill TH2Poly according to a Mercator projection.
gBenchmark->Start("Filling");
for (i=0; i<npoints; i++) {
   longitude = r.Uniform(lon1,lon2);
   latitude  = r.Uniform(lat1,lat2);
   x         = longitude;
   y         = 38*TMath::Log(TMath::Tan(pi4+alpha*latitude));
   p->Fill(x,y);
}
gBenchmark->Show("Filling");

Int_t nbins = p->GetNumberOfBins();
Double_t maximum = p->GetMaximum();


// h2 contains the surfaces computed from TH2Poly.
TH1F *h2 = h->Clone("h2");
h2->Reset();
for (j=0; j<nx; j++) {
   for (i=0; i<nbins; i++) {
      if (strstr(countries[j],p->GetBinName(i+1))) {
         h2->Fill(countries[j],p->GetBinContent(i+1));
         h2->SetBinError(j, p->GetBinError(i+1));
      }
   }
}
```

```
// Normalize the TH2Poly bin contents to the real surfaces.
Double_t scale = surfaces[0]/maximum;
for (i=0; i<nbins; i++) p->SetBinContent(i+1, scale*p->GetBinContent(i+1));

gStyle->SetOptStat(1111);
gStyle->SetPalette(1);
p->Draw("COLZ");
```

。。。。。。

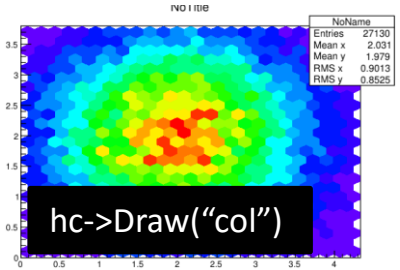
/tutorials/hist/th2polyHoneycomb.C



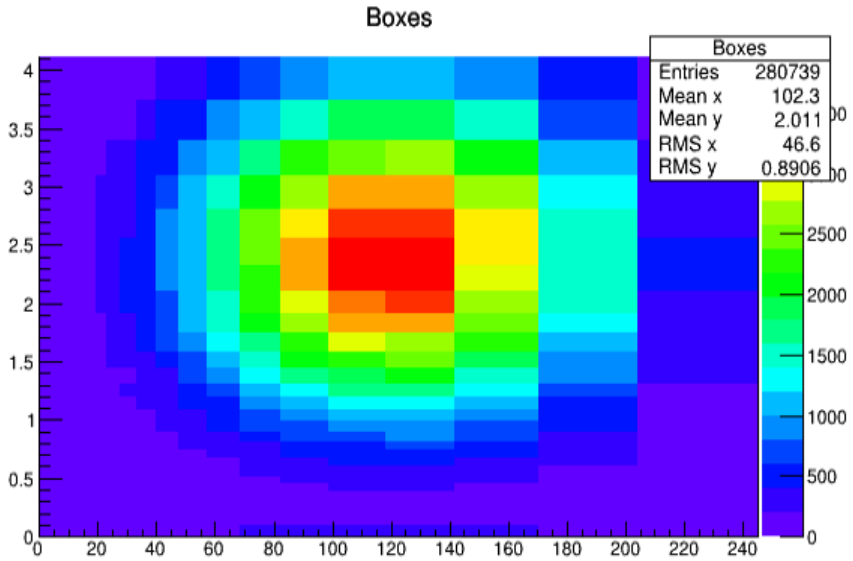


```
//This tutorial illustrates how to create an histogram with hexagonal
//bins (TH2Poly), fill it and draw it using GL.
//
//Author: Olivier Couet

void th2polyHoneycomb(){
   gStyle->SetCanvasPreferGL(true);
   TH2Poly *hc = new TH2Poly();
   hc->Honeycomb(0,0,.1,25,25);
   gStyle->SetPalette(1);

   TRandom ran;
   for (int i = 0; i<30000; i++) {
      hc->Fill(ran.Gaus(2.,1), ran.Gaus(2.,1));
   }

   hc->Draw("gllego");
}
```

Honeycomb(Double_t xstart, Double_t ystart, Double_t a, Int_t k, Int_t s)
a: side length
k: number of the hexagons in raw
s: number of the hexagons in col

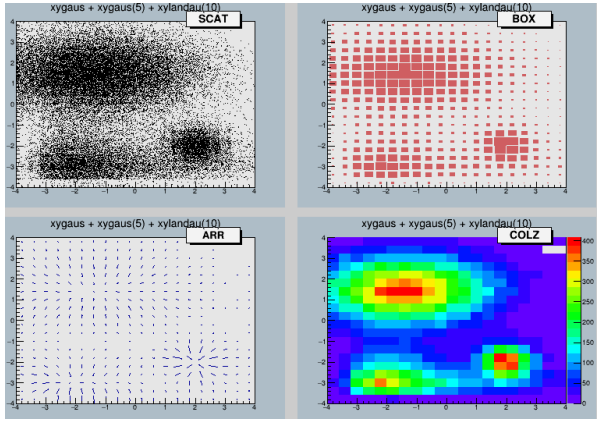


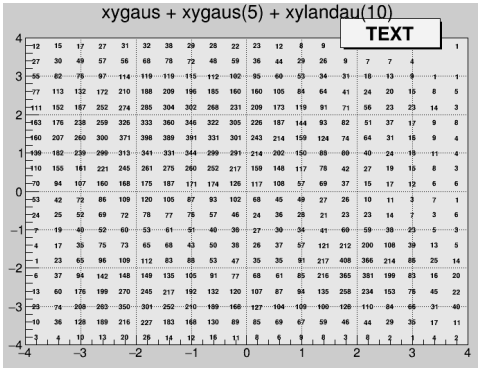
/tutorials/hist/th2polyBoxes.C

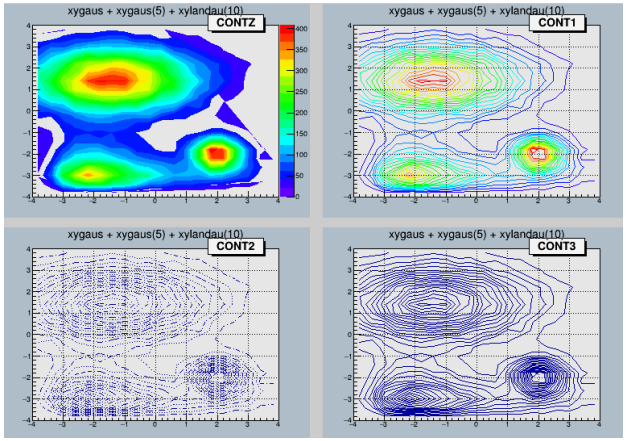


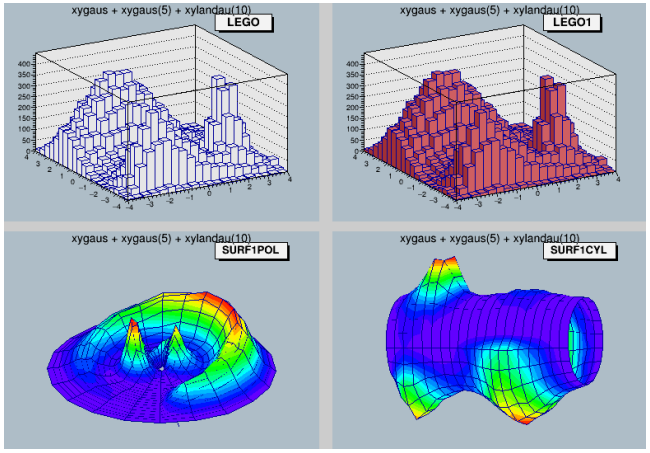

```
{
   TCanvas *ch2p2 = new TCanvas("ch2p2","ch2p2",600,400);
   TH2Poly *h2p = new TH2Poly();
   h2p->SetName("Boxes");
   h2p->SetTitle("Boxes");
   gStyle->SetPalette(1);

   Int_t i,j;
   Int_t nx = 40;
   Int_t ny = 40;
   Double_t x1,y1,x2,y2;
   Double_t dx=0.2, dy=0.1;
   x1 = 0.;
   x2 = dx;

   for (i = 0; i<nx; i++) {
      y1 = 0.;
      y2 = dy;
      for (j = 0; j<ny; j++) {
         h2p->AddBin(x1, y1, x2, y2);
         y1 = y2;
         y2 = y2+y2*dy;
      }
      x1 = x2;
      x2 = x2+x2*dx;
   }

   TRandom ran;
   for (i = 0; i<300000; i++) {
      h2p->Fill(50*ran.Gaus(2.,1), ran.Gaus(2.,1));
   }

   h2p->Draw("COLZ");
   return ch2p2;
}
```

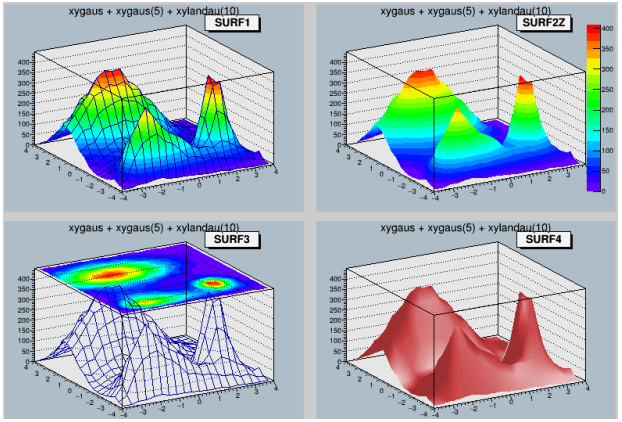
/tutorials/hist/draw2dopt.C







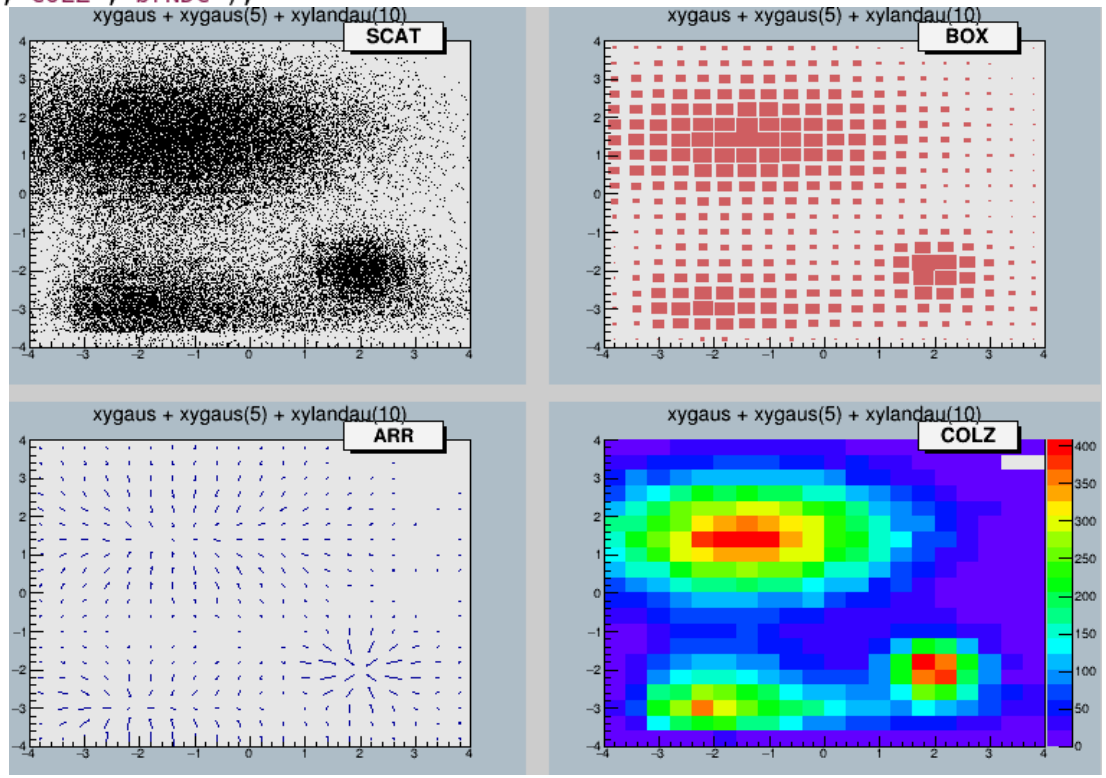




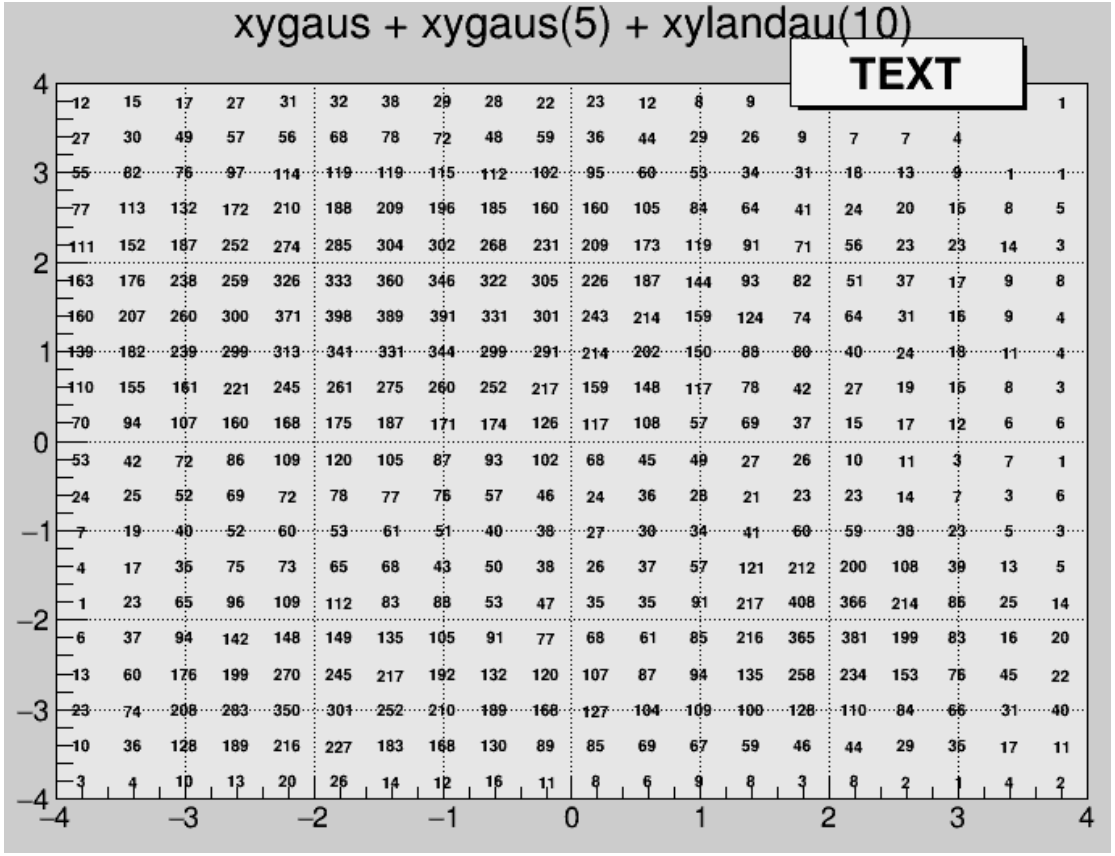
siguang@pku.edu.cn

```
{
   // display the various 2-d drawing options
   //Author: Rene Brun

   gROOT->Reset();
   gStyle->SetOptStat(0);
   gStyle->SetPalette(1);
   gStyle->SetCanvasColor(33);
   gStyle->SetFrameFillColor(18);
   TF2 *f2 = new TF2("f2","xygaus + xygaus(5) + xylandau(10)",-4,4,-4,4);
   Double_t params[] = {130,-1.4,1.8,1.5,1, 150,2,0.5,-2,0.5, 3600,-2,0.7,-3,0.3};
   f2.SetParameters(params);
   TH2F h2("h2","xygaus + xygaus(5) + xylandau(10)",20,-4,4,20,-4,4);
   h2.SetFillColor(46);
   h2.FillRandom("f2",40000);
   TPaveLabel pl;

   //basic 2-d options
   Float_t x1=0.67, y1=0.875, x2=0.85, y2=0.95;
   Int_t cancolor = 17;
   TCanvas c2h("c2h","2-d options",10,10,800,600);
   c2h.Divide(2,2);
   c2h.SetFillColor(cancolor);
```

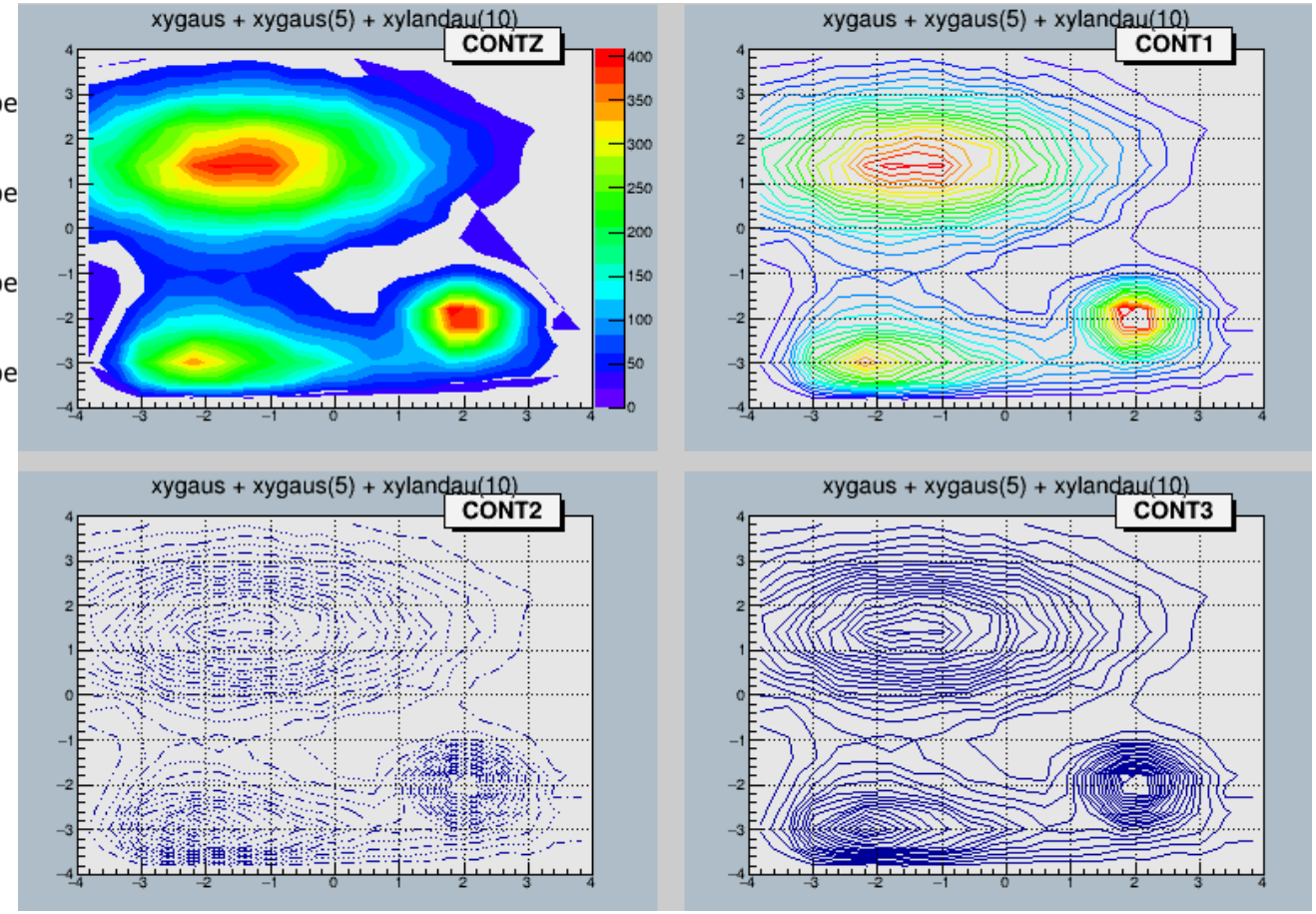
```
c2h.cd(1);
h2.Draw();       pl.DrawPaveLabel(x1,y1,x2,y2,"SCAT","brNDC");
c2h.cd(2);
h2.Draw("box");  pl.DrawPaveLabel(x1,y1,x2,y2,"BOX","brNDC");
c2h.cd(3);
h2.Draw("arr");  pl.DrawPaveLabel(x1,y1,x2,y2,"ARR","brNDC");
c2h.cd(4);
h2.Draw("colz"); pl.DrawPaveLabel(x1,y1,x2,y2,"COLZ","brNDC");
c2h.Update();
```



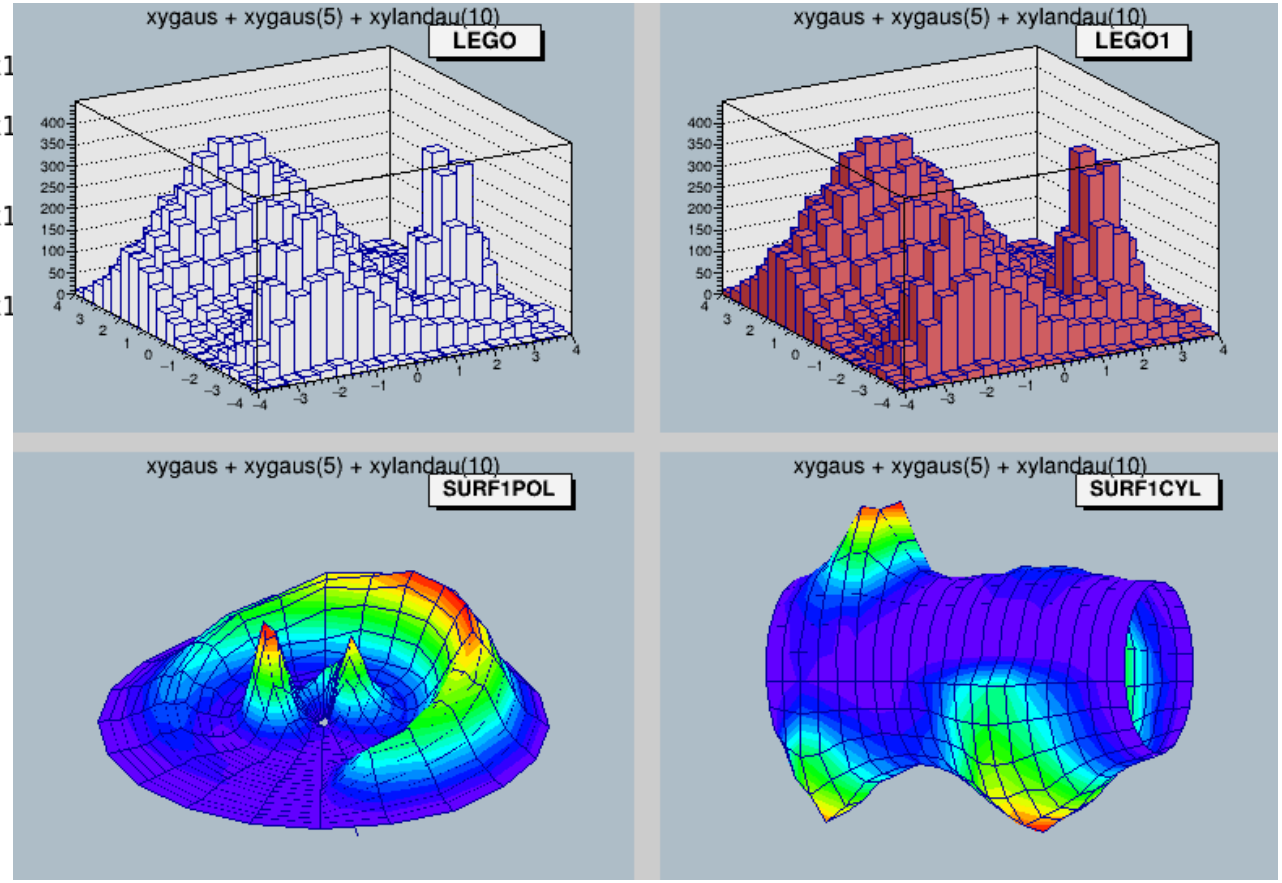
```
//text option
TCanvas ctext("ctext","text option",50,50,800,600);
gPad->SetGrid();
ctext.SetFillColor(cancolor);
ctext->SetGrid();
h2.Draw("text"); pl.DrawPaveLabel(x1,y1,x2,y2,"TEXT","brNDC");
ctext.Update();
```



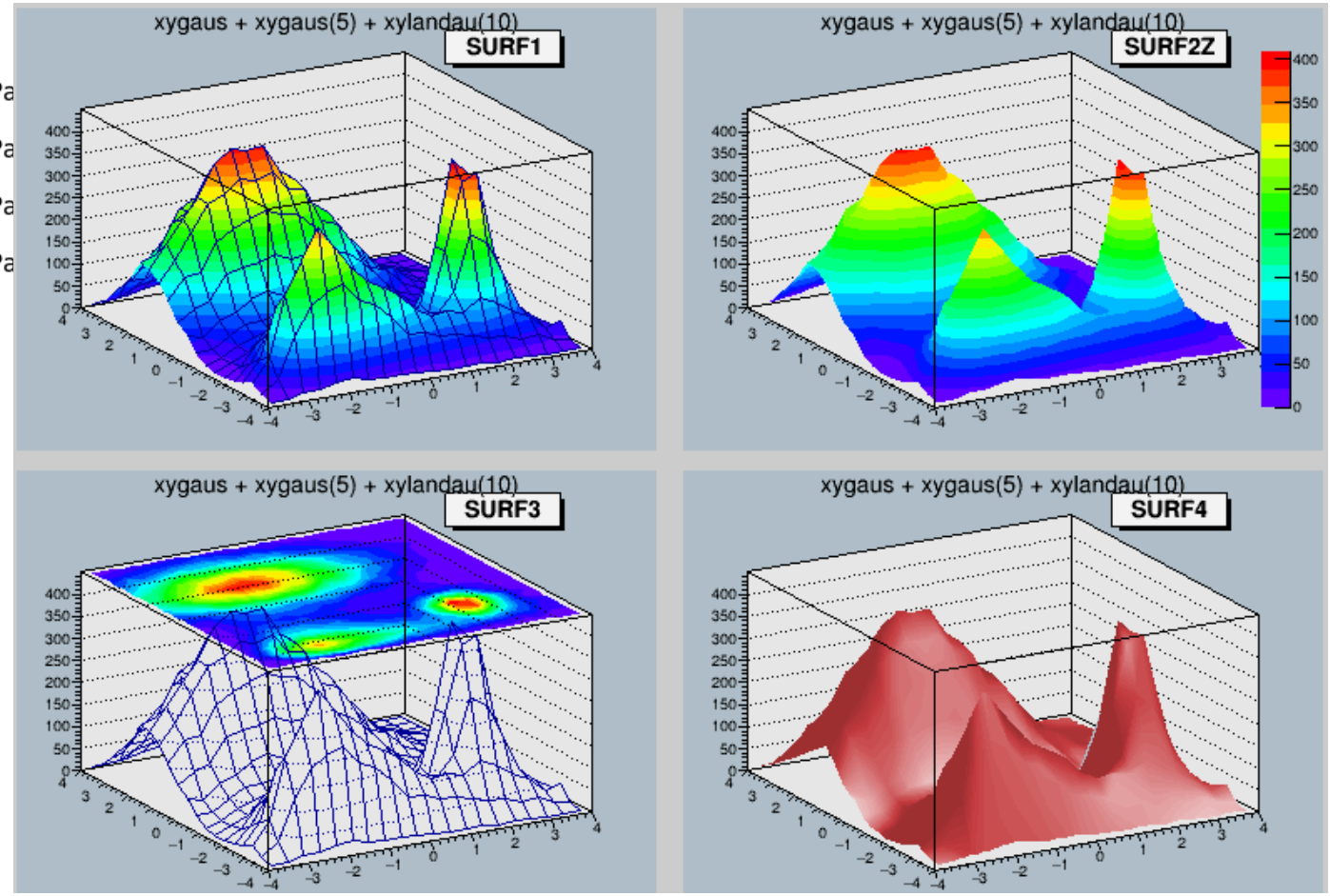
```
//contour options
TCanvas cont("contours","contours",100,100,800,600);
cont.Divide(2,2);
gPad->SetGrid();
cont.SetFillColor(cancolor);
cont.cd(1);
h2.Draw("contz"); pl.DrawPaveLabe
cont.cd(2);
gPad->SetGrid();
h2.Draw("cont1"); pl.DrawPaveLabe
cont.cd(3);
gPad->SetGrid();
h2.Draw("cont2"); pl.DrawPaveLabe
cont.cd(4);
gPad->SetGrid();
h2.Draw("cont3"); pl.DrawPaveLabe
cont.Update();
```



```
//lego options
TCanvas lego("lego","lego options",150,150,800,600);
lego.Divide(2,2);
lego.SetFillColor(cancolor);
lego.cd(1);
h2.Draw("lego");      pl.DrawPaveLabel(x1
lego.cd(2);
h2.Draw("lego1");     pl.DrawPaveLabel(x1
lego.cd(3);
gPad->SetTheta(61); gPad->SetPhi(-82);
h2.Draw("surf1pol"); pl.DrawPaveLabel(x1
lego.cd(4);
gPad->SetTheta(21); gPad->SetPhi(-90);
h2.Draw("surf1cyl"); pl.DrawPaveLabel(x1
lego.Update();
```



```
//surface options
TCanvas surf("surfopt","surface options",200,200,800,600);
surf.Divide(2,2);
surf.SetFillColor(cancolor);
surf.cd(1);
h2.Draw("surf1");    pl.DrawPa
surf.cd(2);
h2.Draw("surf2z");   pl.DrawPa
surf.cd(3);
h2.Draw("surf3");    pl.DrawPa
surf.cd(4);
h2.Draw("surf4");    pl.DrawPa
surf.Update();
```



A ROOT Guide For Beginners

来自：

https://root.cern.ch/drupal/content/users-guide#primer

2.1 ROOT as calculator

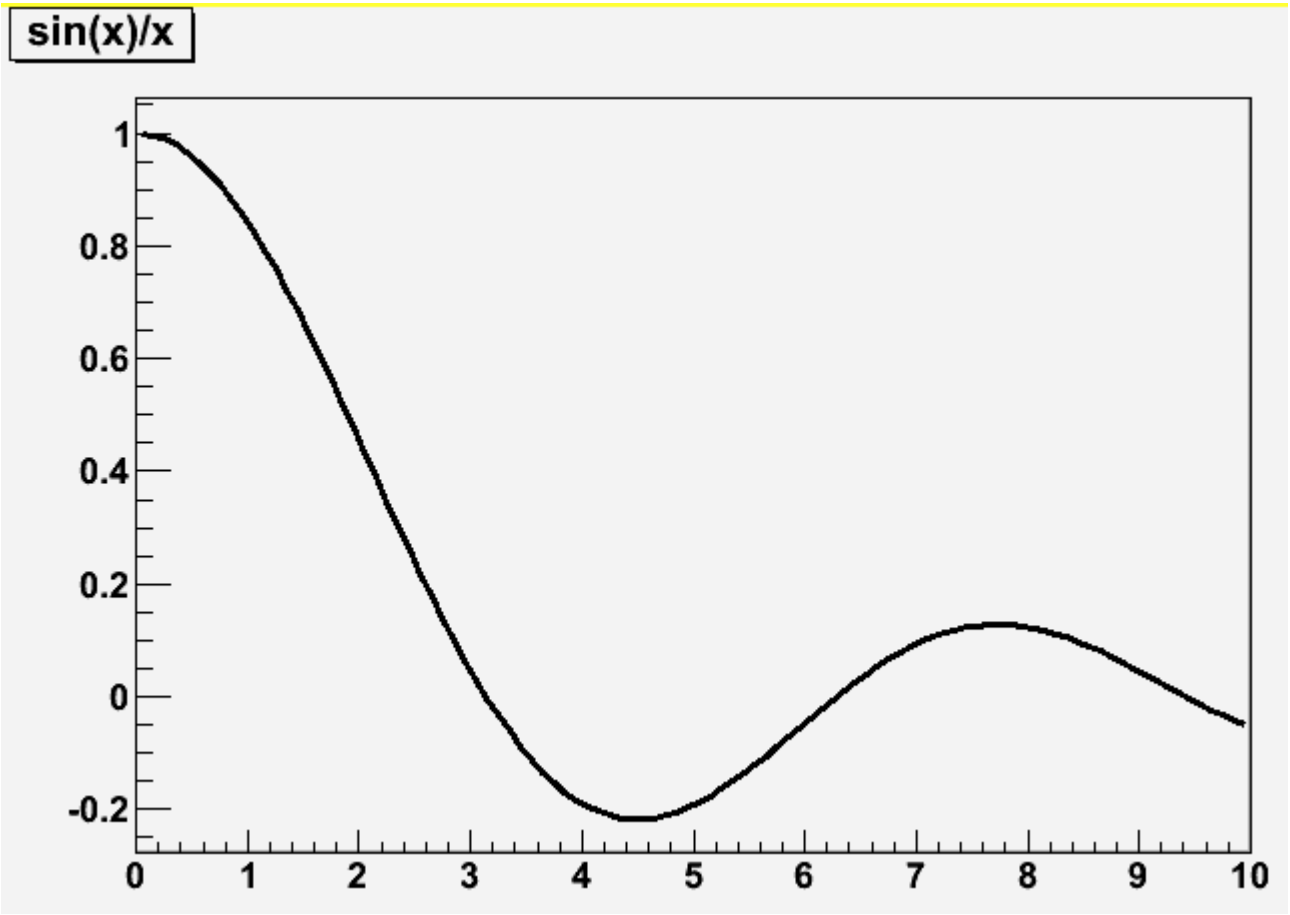
```
root [0] 1+1
(const int)2
root [1] 2*(4+2)/12.
(const double)1.00000000000000e+00
root [2] sqrt(3)
(const double)1.73205080756887719e+00
root [3] 1 > 2
(const int)0
root [4] TMath::Pi()
(Double_t)3.14159265358979312e+00
root [5] TMath::Erf(.2)
(Double_t)2.22702589210478447e-01
```

ROOT as calculator [Continue]

```
root [6] double x=.5
root [7] int N=30
root [8] double geom_series=0
root [9] for (int i=0;i<N;++i)geom_series+=TMath::Power(x,i)
root [10] TMath::Abs(geom_series - (1-TMath::Power(x,N-1))/(1-x))
(Double_t)1.8626451492309570 3e-09
```

2.2 ROOT as Function Plotter

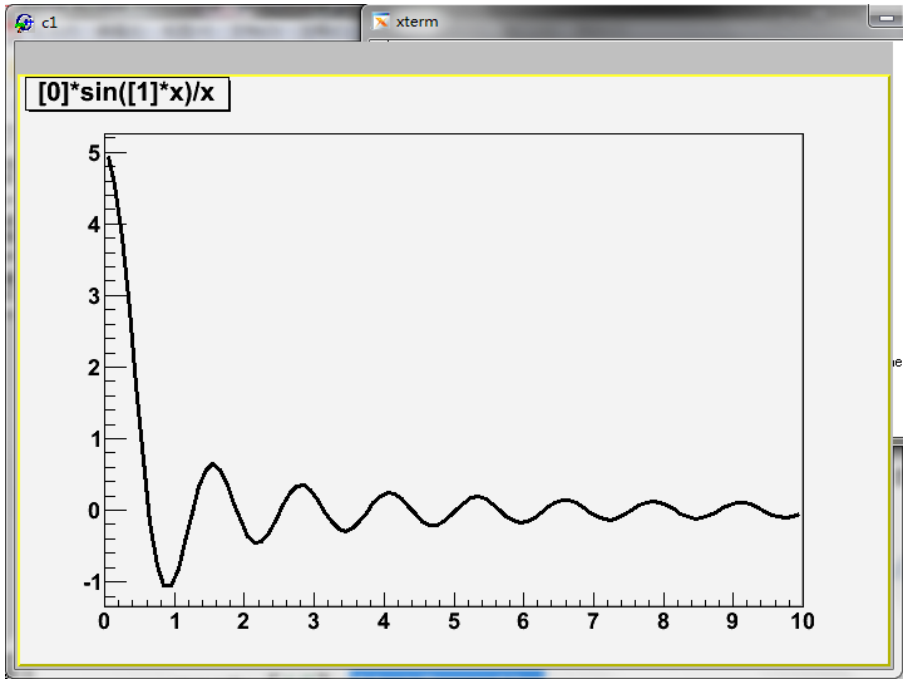
```
root [11] TF1 *f1 = new TF1("f1","sin(x)/x",0.,10.);
root [12] f1->Draw();
```



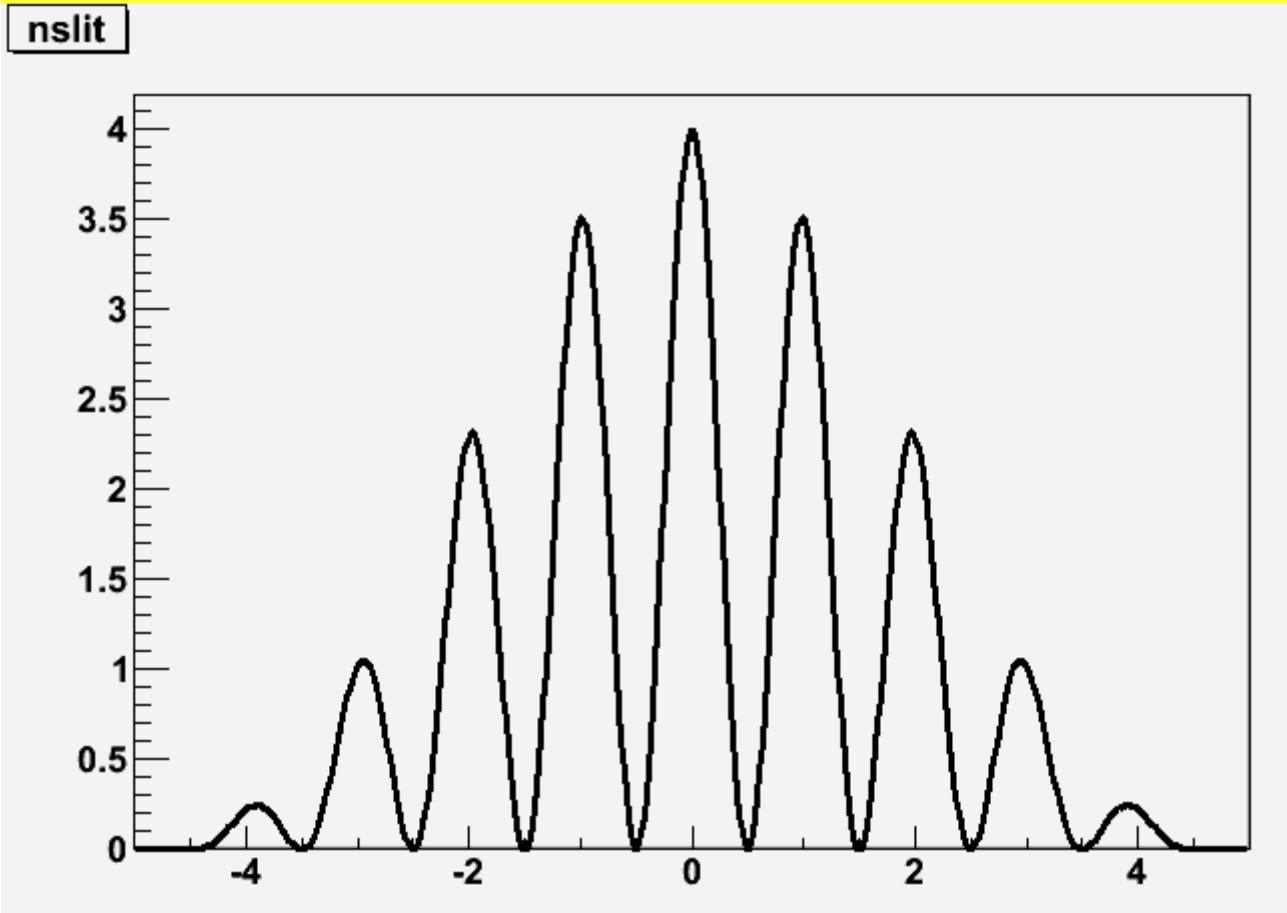
ROOT as Function Plotter [Continue]

```
root [13] TF1 *f1 = new TF1("f2","[0]*sin([1]*x)/x",0.,10.);
root [14] f1->SetParameter(0,1);
root [15] f1->SetParameter(1,1);
root [16] f1->Draw();
```

root [] f1->SetParameter(1,5);
root [] f1->Draw()



写成代码运行



代码名： slits.C

slits.C

```
// Example drawing the interference pattern of light
// falling on a grid with n slits and ratio r of slit
// width over distance between slits.

// function code in C
double single(double *x, double *par) {
  double const pi=4*atan(1.);
  return pow(sin(pi*par[0]*x[0])/(pi*par[0]*x[0]),2);
}

double nslit0(double *x,double *par){
  double const pi=4*atan(1.);
  return pow(sin(pi*par[1]*x[0])/sin(pi*x[0]),2);
}

double nslit(double *x, double *par){
  return single(x,par) * nslit0(x,par);
}

// This is the main program
void slits() {
  float r,ns;
  // request user input
  cout << "slit width: r=0.2? ";
  scanf("%f",&r);
  cout << "# of slits ns=2?";
  scanf("%f",&ns);
  cout <<"interference pattern for "<< ns
       <<" slits, width/distance: "<<r<<endl;

  // define function and set options
  TF1 *Fnslit = new TF1("Fnslit",nslit,-5.001,5.,2);
  Fnslit->SetNpx(500);
```

slits.C [continue]

```
  // set parameters, as read in above
  Fnslit->SetParameter(0,r);
  Fnslit->SetParameter(1,ns);

  // draw the interference pattern for a grid with n slits
  Fnslit->Draw();
}
```

练习：

1）在上面画箭头然后保存成c1.C文件；保存成.eps 等文件

2）增加坐标轴的标题

Controlling ROOT

在root[]输入如下命令看看啥反应？

.q

.?

.!ls

.!pwd

.x slits.C　要在当前目录下有这个文件奥

.L slits.C

slits()　记不全打tab键看看

要想编译运行，slits.C加头文件

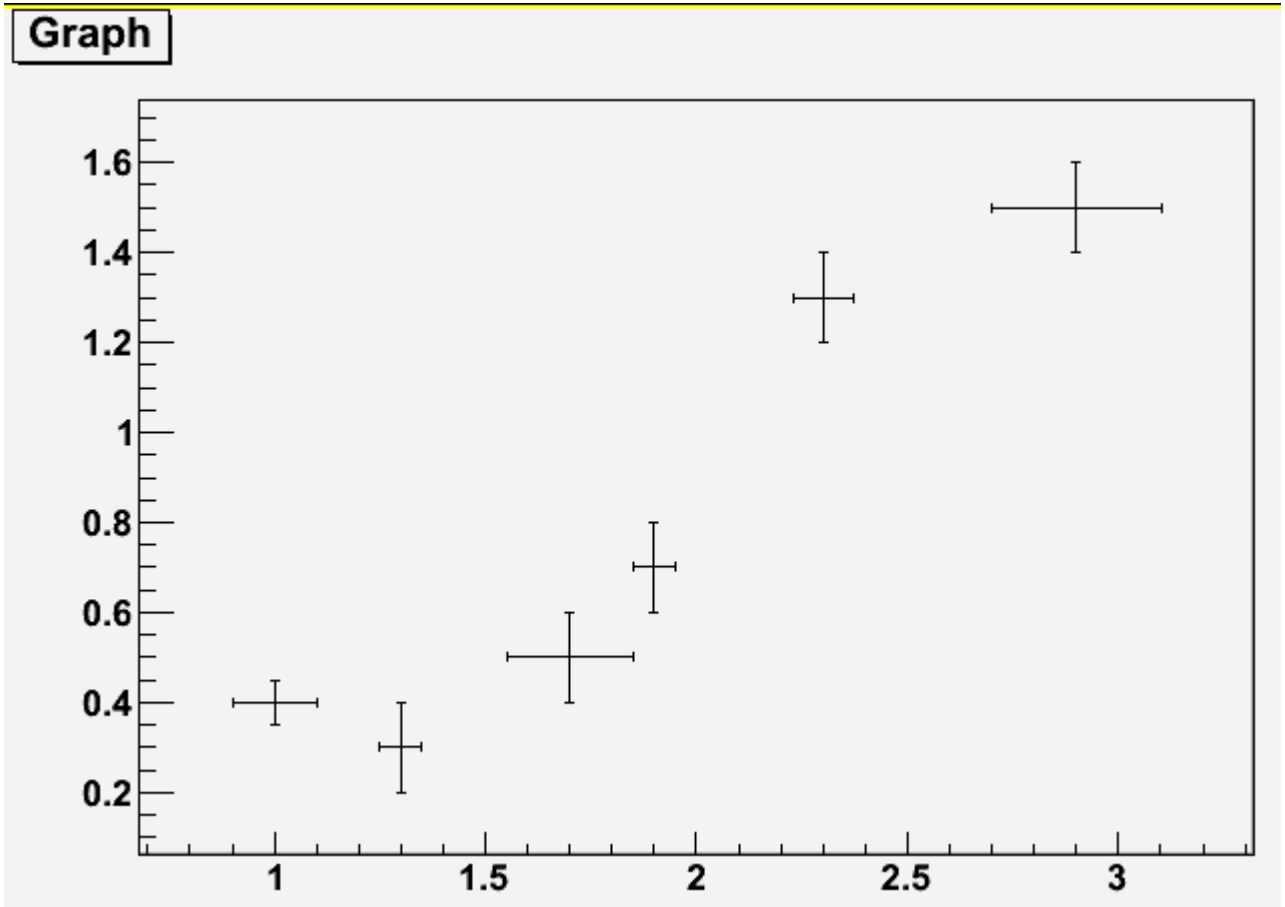
```
#include <iostream>
#include <TF1.h>

using namespace std;
```

后

.L slits.C++

slits()

Plotting Measurements



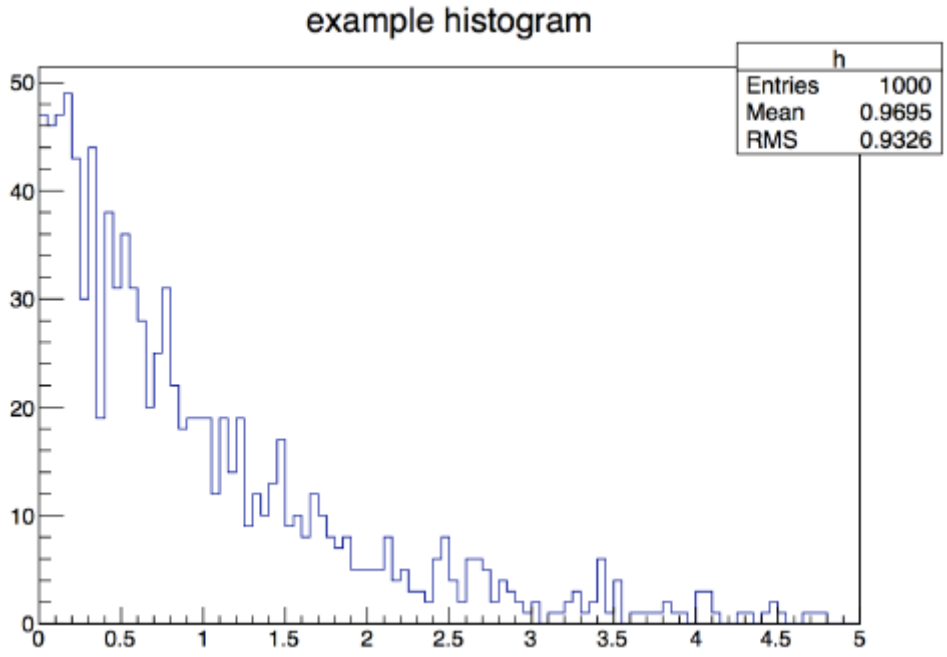

```
root [0] TGraphErrors *gr=new TGraphErrors("ExampleData.txt")
root [1] gr->Draw("AP")
```

ExampleData.txt

```
[] cat ExampleData.txt
# fake data to demonstrate the use of TGraphErrors

# x y ex ey
  1. 0.4 0.1 0.05
  1.3 0.3 0.05 0.1
  1.7 0.5 0.15 0.1
  1.9 0.7 0.05 0.1
  2.3 1.3 0.07 0.1
  2.9 1.5 0.2 0.1
```

2.5 Histograms in ROOT

```
root [0] TF1 efunc("efunc","exp([0]+[1]*x)",0.,5.);
root [1] efunc.SetParameter(0,1);
root [2] efunc.SetParameter(1,-1);
root [3] TH1F* h=new TH1F("h","example histogram",100,0.,5.);
root [4] for (int i=0;i<1000;i++) {h->Fill(efunc.GetRandom());}
root [5] h->Draw();
```



读文件填充

```
root [1] TH1F* h=new TH1F("h","example histogram",100,0.,5.);
root [2] ifstream inp; double x;
root [3] inp.open("expo.dat");
root [4] while (inp >> x) { h->Fill(x); }
root [5] h->Draw();
root [6] inp.close();
```

expo.data
为每行一个数的文本文件

在Tcanvas上用鼠标右键看看有啥反应？！
可以设线，点的属性，坐标轴的Title
等。。。。。

练练吧！

rootlogon.C

```
[hepfarm02]  cat $ROOTSYS/tutorials/rootlogon.C
{
   printf("\nWelcome to the ROOT tutorials\n\n");
   printf("\nType \".x demos.C\" to get a toolbar from which to execute the demos\n");
   printf("\nType \".x demoshelp.C\" to see the help window\n\n");
   printf("==> Many tutorials use the file hsimple.root produced by hsimple.C\n");
   printf("==> It is recommended to execute hsimple.C before any other script\n\n");
}
```

rootlogon.C

```
// This is the file rootlogon.C
{
   TStyle *myStyle  = new TStyle("MyStyle","My Root Styles");

   // from ROOT plain style
   myStyle->SetCanvasBorderMode(0);
   myStyle->SetPadBorderMode(0);
   myStyle->SetPadColor(0);
   myStyle->SetCanvasColor(0);
   myStyle->SetTitleColor(1);
   myStyle->SetStatColor(0);

   myStyle->SetLabelSize(0.03,"xyz"); // size of axis values

   // default canvas positioning
   myStyle->SetCanvasDefX(900);
   myStyle->SetCanvasDefY(20);
   myStyle->SetCanvasDefH(550);
   myStyle->SetCanvasDefW(540);

   myStyle->SetPadBottomMargin(0.1);
   myStyle->SetPadTopMargin(0.1);
   myStyle->SetPadLeftMargin(0.1);
   myStyle->SetPadRightMargin(0.1);
   myStyle->SetPadTickX(1);
   myStyle->SetPadTickY(1);
   myStyle->SetFrameBorderMode(0);

   // Din letter
   myStyle->SetPaperSize(21, 28);
```

rootlogon.C [continue]

```
    myStyle->SetOptStat(111111);// Show overflow and underflow as well
    myStyle->SetOptFit(1011);
    myStyle->SetPalette(1);

    // apply the new style
    gROOT->SetStyle("MyStyle"); //uncomment to set this style
    gROOT->ForceStyle(); // use this style, not the one saved in root files

    printf("\n Beginning new ROOT session with private TStyle \n");

}
```

.rootrc

实现方式，按顺序检查：

- .rootrc //local directory
- $HOME/.rootrc //user directory
- $ROOTSYS/etc/system.rootrc //global ROOT directory

内有

```
# Rint (interactive ROOT executable) specific alias, logon and logoff macros.
Rint.Load:              rootalias.C
Rint.Logon:             rootlogon.C
Rint.Logoff:            rootlogoff.C
```

2.7.3 ROOT command history

```
cat ~/.root_hist
```

俺喜欢的

```
#include <TStyle.h>

// Set the general style options
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); not In root5.28
  gStyle->SetFillStyle(0);
}
```

用法：

1)在一个文件中例如useful.h敲入左边代码

2) emacs testStyle.C &

```
#include "useful.h"
void testStyle(){
  TH1F *h1 = new TH1F("h1","",100,-10,10);
  SetSgStyle();
  TH1F *h2 = new TH1F("h2","",100,-10,10);
  h1->FillRandom("gaus",1000);
  h2->FillRandom("gaus",1000);
  TCanvas *c1 = new TCanvas("c1","");
  c1->Divide(2,1);
  c1->cd(1);
  h1->Draw();
  c1->cd(2);
  h2->Draw();
}
```

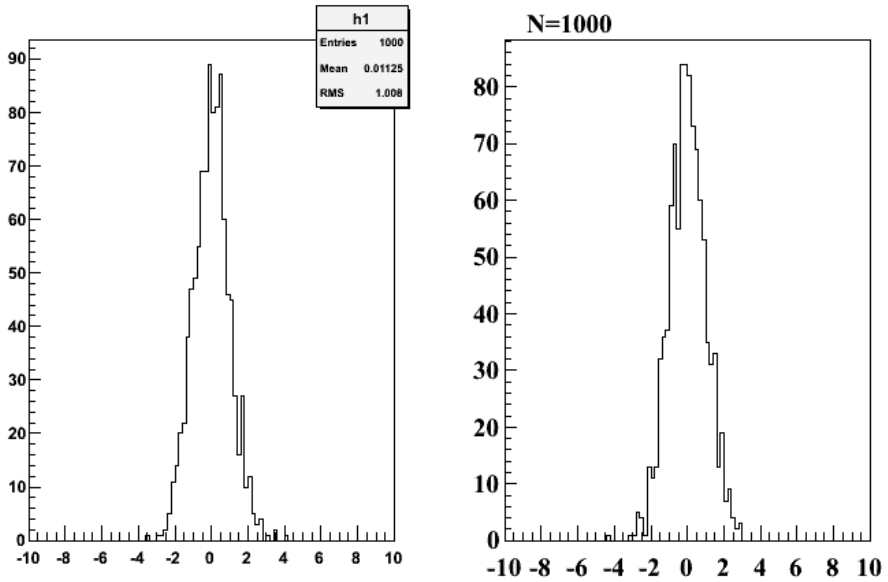
txtN

在useful.h中加入：

```
void txtN(Double_t x0, Double_t y0,TH1 *h,Char_t sName[]="N=%.0f",Double_t sizeTxt=0.06){
  h->SetStats(kFALSE);
  TLatex *ltx = new TLatex();
  ltx->SetNDC(kTRUE);
  ltx->SetTextColor(h->GetLineColor());
  ltx -> SetTextFont(22);
  ltx->SetTextSize(sizeTxt);
  ltx->DrawLatex(x0,y0,Form(sName,h->GetEntries()));
  gPad->Modified();
  gPad->Update();
}
```

在testStyle.C中加入

```
txtN(0.2,0.95,h2);
```



newTH1F

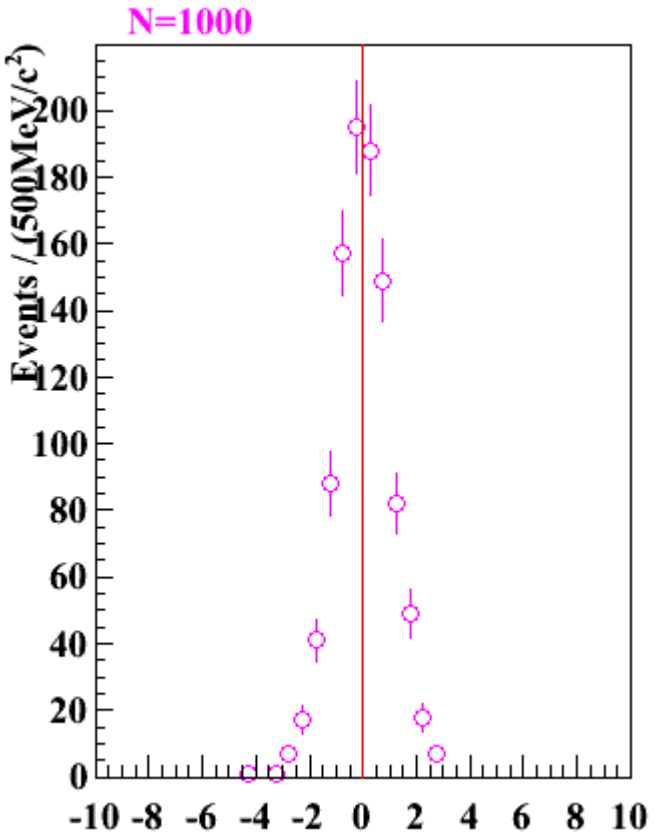
```
TH1F * newTH1F(Char_t name[]="h1",Double_t binw=0.01, Double_t LowBin=0.0, Double_t HighBin=3.0,Bo
_t MevTitle = kTRUE,Int_t iMode=-1){
  Int_t nbin = TMath::Nint( (HighBin - LowBin)/binw );
  HighBin = binw*nbin + LowBin;

  TH1F *h = new TH1F(name,"",nbin,LowBin,HighBin);
  if(MevTitle) h->GetYaxis()->SetTitle(Form("Events / (%.0fMeV/c^{2})",h->GetBinWidth(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;
}
```

```
void testStyle2(){
  TH1F *h1 = newTH1F("h1",0.5,-10,10,kTRUE,1);
  SetSgStyle();
  TH1F *h2 = newTH1F("h2",0.5,-10,10,kTRUE,2);
  h1->FillRandom("gaus",1000);
  h2->FillRandom("gaus",1000);
  TCanvas *c1 = new TCanvas("c1","");
  c1->Divide(2,1);
  c1->cd(1);
  h1->Draw("EP");
  c1->cd(2);
  h2->Draw("EP");
  txtN(0.2,0.95,h2);
}
```

LineX1

```
void LineX1(Double_t atX,Int_t iColor=kRed,Int_t iStyle=1,Double_t iWidth=1){
  gPad->Modified();
  gPad->Update();
  TLine *l1 = new TLine(atX,gPad->GetUymin(),atX,gPad->GetUymax());
  l1->SetLineColor(iColor);
  l1->SetLineStyle(iStyle);
  l1->SetLineWidth(iWidth);
  l1->Draw();
}
```

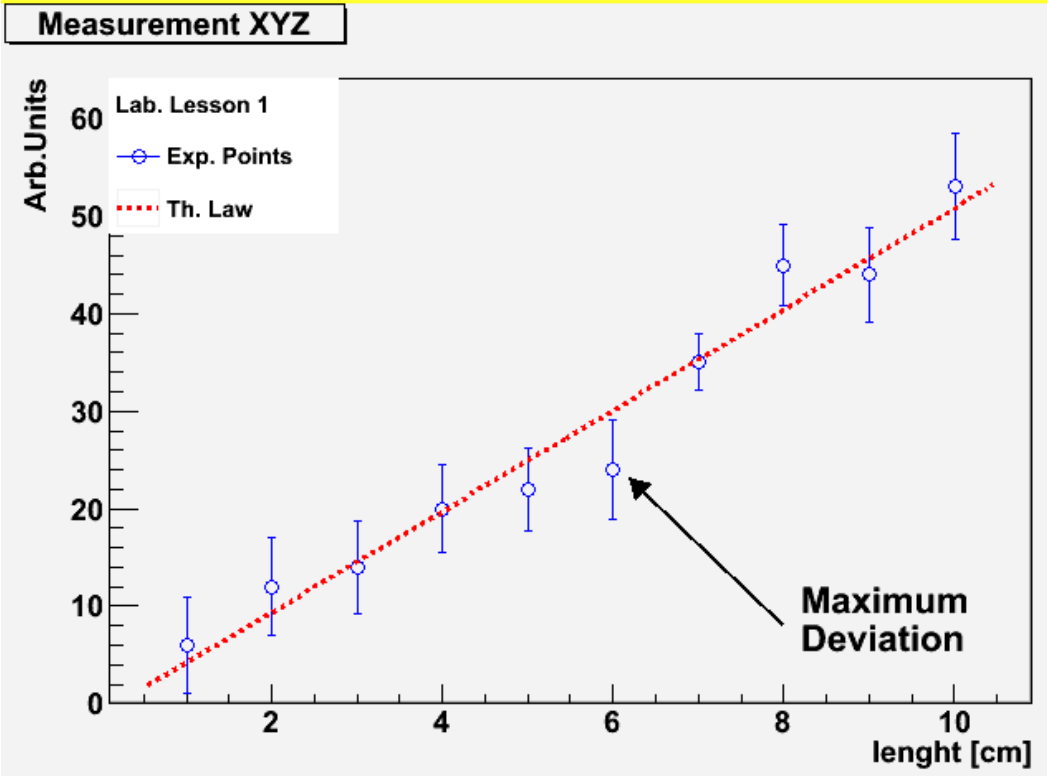
```
void testStyle2(){
  TH1F *h1 = newTH1F("h1",0.5,-10,10,kTRUE,1);
  SetSgStyle();
  TH1F *h2 = newTH1F("h2",0.5,-10,10,kTRUE,2);
  h1->FillRandom("gaus",1000);
  h2->FillRandom("gaus",1000);
  TCanvas *c1 = new TCanvas("c1","");
  c1->Divide(2,1);
  c1->cd(1);
  h1->Draw("EP");
  c1->cd(2);
  h2->Draw("EP");
  txtN(0.2,0.95,h2);
  LineX1(0.0);
}
```



避免多次引用保护

```
#ifndef USEFUL_H
#define USEFUL_H
#include <TStyle.h>

// Set the general style options
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");


...

...

...


#endif
```

3.2 A more complete example



siguang@pku.edu.cn

Marco1.C

```
// Builds a graph with errors, displays it and saves it as
// image. First, include some header files (within CINT,
// these will be ignored).

#include "TCanvas.h"
#include "TROOT.h"
#include "TGraphErrors.h"
#include "TF1.h"
#include "TLegend.h"
#include "TArrow.h"
#include "TLatex.h"

void macro1(){
    // The values and the errors on the Y axis
    const int n_points=10;
    double x_vals[n_points]=
            {1,2,3,4,5,6,7,8,9,10};
    double y_vals[n_points]=
            {6,12,14,20,22,24,35,45,44,53};
    double y_errs[n_points]=
            {5,5,4.7,4.5,4.2,5.1,2.9,4.1,4.8,5.43};

    // Instance of the graph
    TGraphErrors graph(n_points,x_vals,y_vals,NULL,y_errs);
    graph.SetTitle("Measurement XYZ;lenght [cm];Arb.Units");

    // Make the plot estetically better
    graph.SetMarkerStyle(kOpenCircle);
    graph.SetMarkerColor(kBlue);
    graph.SetLineColor(kBlue);

    // The canvas on which we'll draw the graph
    TCanvas* mycanvas = new TCanvas();

    // Draw the graph !
    graph.DrawClone("APE");
```

Marco1.C [continue]

```
    // Define a linear function
    TF1 f("Linear law","[0]+x*[1]",.5,10.5);
    // Let's make the funcion line nicer
    f.SetLineColor(kRed); f.SetLineStyle(2);
    // Fit it to the graph and draw it
    graph.Fit(&f);
    f.DrawClone("Same");

    // Build and Draw a legend
    TLegend leg(.1,.7,.3,.9,"Lab. Lesson 1");
    leg.SetFillColor(0);
    graph.SetFillColor(0);
    leg.AddEntry(&graph,"Exp. Points");
    leg.AddEntry(&f,"Th. Law");
    leg.DrawClone("Same");

    // Draw an arrow on the canvas
    TArrow arrow(8,8,6.2,23,0.02,"|>");
    arrow.SetLineWidth(2);
    arrow.DrawClone();

    // Add some text to the plot
    TLatex text(8.2,7.5,"#splitline{Maximum}{Deviation}");
    text.DrawClone();

    mycanvas->Print("graph_with_law.pdf");
}

#ifndef __CINT__
int main(){
    macro1();
    }
#endif
```

Interpretation and Compilation

3.4.1 Compile a Macro with ACLiC (The Automatic Compiler of Libraries for CINT--C/C++ interpreter)

```
root [0] .L macro1.C++
root [1] macro1()
```

3.4.2 Compile a Macro with the Compiler

- **g++ -o macro1.exe macro1.C `root-config --cflags --libs`**
- **./macro1.exe**

- **acroread graph_with_law.pdf &**

ExampleMarco.C

```
/*
 * Note that this file can be either used as a compiled program
    or as a ROOT macro.
 * If it is used as a compiled program, additional include statements
   and the definition of the main program have to be made. This is
   not needed if the code is executed at the ROOT prompt.
                                                                    */

//#ifndef __CINT__       // These include-statements are needed if the program is
#include "TFile.h"      // run as a "stand-alone application", i.e. if it is not
#include "TH1F.h"       // called from an interactive ROOT session.
#include "TCanvas.h"
#include "TMath.h"
// eventually, load some C libraries
#include <math.h>

void ExampleMacro();

//_____________________________________________________________________________
int main()
{
  ExampleMacro();
  return 0;
}
//#endif
```

ExampleMarco.C [continue]

```
//_____________________________________________________________________

/*
   * From here on, the code can also be used as a macro
   * Note though, that CINT may report errors where there are none
     in C++. E.g. this happens here where CINT says that f1 is
     out of scope ...

     ==>>  put your code here
      (remember to update the name of you Macro in the
       lines above if you intend to comile the code)
                                                                    */
void ExampleMacro() {
  // Create a histogram, fill it with random gaussian numbers
  TH1F *h = new TH1F ("h", "example histogram", 100, -5.,5.);
  h->FillRandom("gaus",1000);

  // draw the histogram
  h->DrawClone();

/* - Create a new ROOT file for output
   - Note that this file may contain any kind of ROOT objects, histograms,
     pictures, graphics objects etc.
   - the new file is now becoming the current directory */
  TFile *f1 = new TFile("ExampleMacro.root","RECREATE","ExampleMacro");

  // write Histogram to current directory (i.e. the file just opened)
  h->Write();

  // Close the file.
  //   (You may inspect your histogram in the file using the TBrowser class)
  f1->Close();
}
```

Compile&Run ExampleMarco.C [continue]

```
>g++ -o ExampleMacro.exe ExampleMacro.C `root-config --cflags --libs`
> ./ExampleMacro.exe
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
```

ExampleMacro_GUI.C

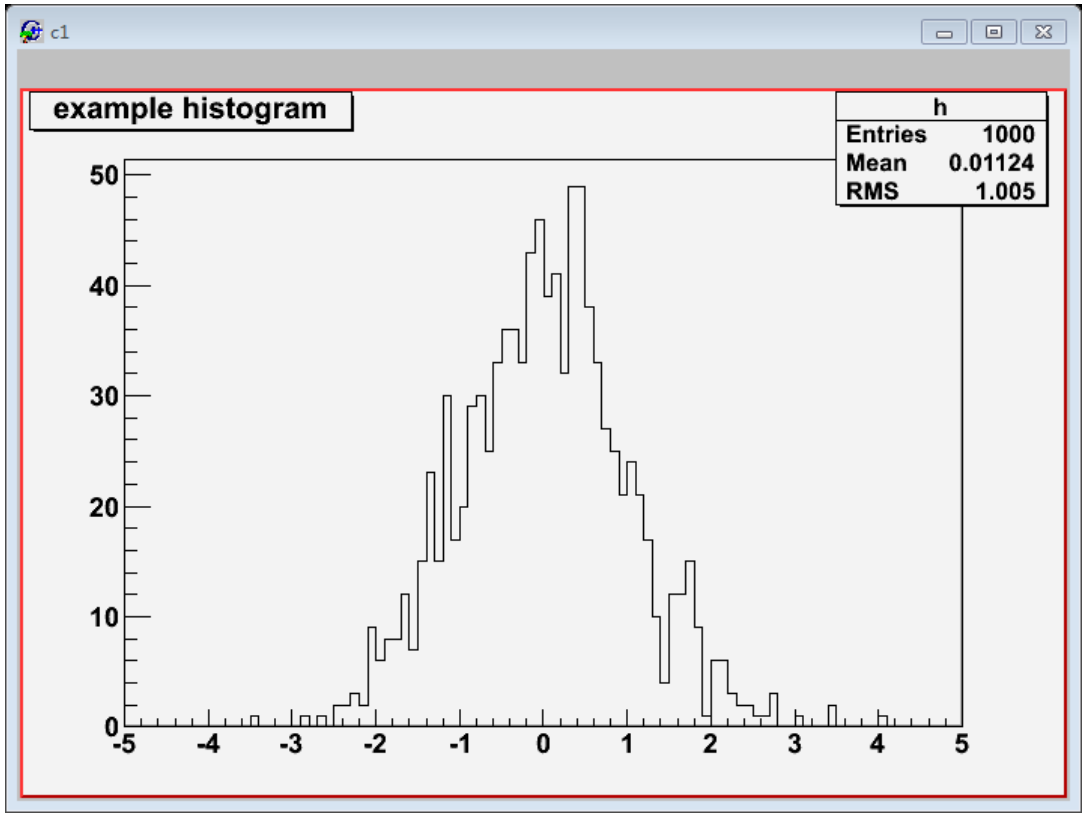
```
/*
 This piece of code demonstrates how a root macro is used as a standalone
    application with full acces the grapical user interface (GUI) of ROOT   */

// include ALL header files needed
#ifndef __CINT__
#include "TROOT.h"
#include "TApplication.h"
#include "TBrowser.h"
#include "TFile.h"
#include "TH1F.h"
#include "TCanvas.h"
#include "TMath.h"
#endif
// eventually, include some additoinal C or C++ libraries
#include <math.h>

//     ==>>  put the code of your macro here
void ExampleMacro_GUI() {
  // Create a histogram, fill it with random gaussian numbers
  TH1F *h = new TH1F ("h", "example histogram", 100, -5.,5.);
  h->FillRandom("gaus",1000);

  // draw the histogram
  h->DrawClone();
```

ExampleMacro_GUI.C [continue]

```
/* - Create a new ROOT file for output
   - Note that this file may contain any kind of ROOT objects, histograms,
     pictures, graphics objects etc.
   - the new file is now becoming the current directory */
  TFile *f1 = new TFile("ExampleMacro.root","RECREATE","ExampleMacro");

  // write Histogram to current directory (i.e. the file just opened)
  h->Write();

  // Close the file.
  //   (You may inspect your histogram in the file using the TBrowser class)
  f1->Close();
}

// the "dressing" code for a stand-alone ROOT application starts here
#ifndef __CINT__
void StandaloneApplication(int argc, char** argv) {
  // ==>> here the ROOT macro is called
  ExampleMacro_GUI();
}

// This is the standard main of C++ starting a ROOT application
int main(int argc, char** argv) {
   TApplication app("Root Application", &argc, argv);
   StandaloneApplication(app.Argc(), app.Argv());
   app.Run();
   return 0;
}
#endif
```

Compile&Run ExampleMarco_GUI.C [continue]

```
>g++ -o ExampleMacro_GUI.exe ExampleMacro_GUI.C `root-config --cflags --libs`
> ./ExampleMacro_GUI.exe
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
```



Read Graph Points from File

```
TGraphErrors(const char *filename,
const char *format="%lg %lg %lg %lg", Option_t *option="");
```

The format string can be:

- "\%lg \%lg" read only 2 first columns into X,Y
- "\%lg \%lg \%lg" read only 3 first columns into X,Y and EY
- "\%lg \%lg \%lg \%lg" read only 4 first columns into X,Y,EX,EY

macro2.C

macro2_input.txt

```
# Measurement of Friday 26 March
# Experiment 2 Physics Lab

1   6   5
2   12  5
3   14  4.7
4   20  4.5
5   22  4.2
6   24  5.1
7   35  2.9
8   45  4.1
9   44  4.8
10  53  5.43
```

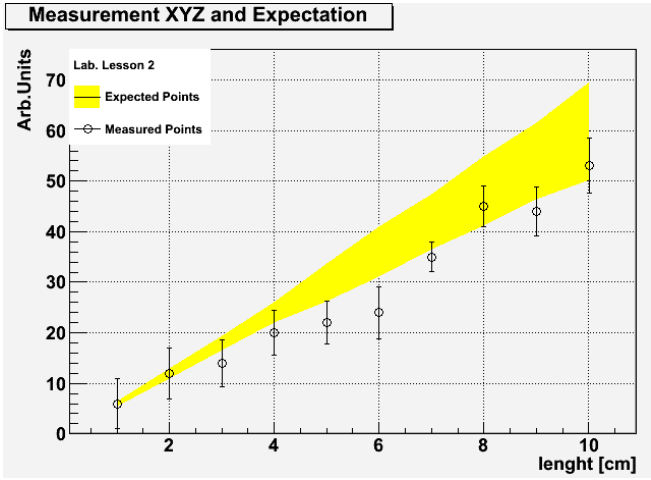
macro2_input_expected.txt

```
# Measurement of Friday 26 March
# Experiment 2 Physics Lab
# Expected points from theory predictions

1   6   0.5
2   12  1.
3   18  1.5
4   24  2.0
5   30  3.7
6   36  4.9
7   42  5.4
8   48  6.8
9   54  7.5
10  60  9.7
```

marco2.C

```
// Reads the points from a file and produces a simple graph.
int macro2(){
    TCanvas* c=new TCanvas();
    c->SetGrid();

    TGraphErrors graph_expected("./macro2_input_expected.txt",
                                "%lg %lg %lg");
    graph_expected.SetTitle(
       "Measurement XYZ and Expectation;
        lenght [cm];
        Arb.Units");
    graph_expected.SetFillColor(kYellow);
    graph_expected.DrawClone("E3AL"); // E3 draws the band

    TGraphErrors graph("./macro2_input.txt","%lg %lg %lg");
    graph.SetMarkerStyle(kCircle);
    graph.SetFillColor(0);
    graph.DrawClone("PESame");

    // Draw the Legend
    TLegend leg(.1,.7,.3,.9,"Lab. Lesson 2");
    leg.SetFillColor(0);
    leg.AddEntry(&graph_expected,"Expected Points");
    leg.AddEntry(&graph,"Measured Points");
    leg.DrawClone("Same");

    graph.Print();
}
```

graph.Print()显示数据信息

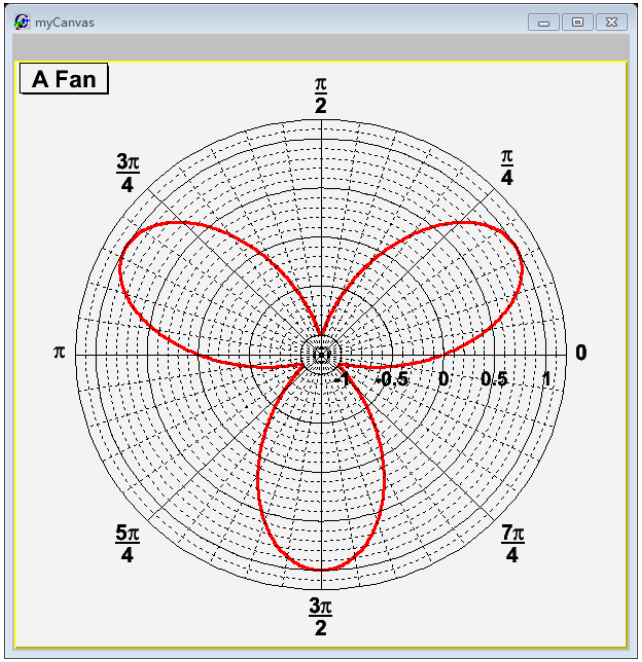


siguang@pku.edu.cn

Polar Graphs

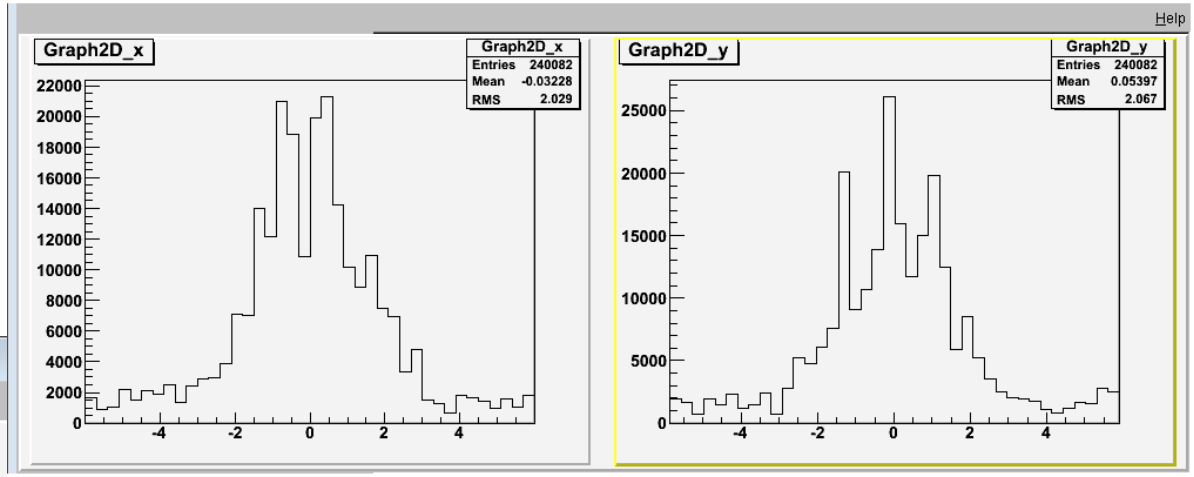
macro3.C

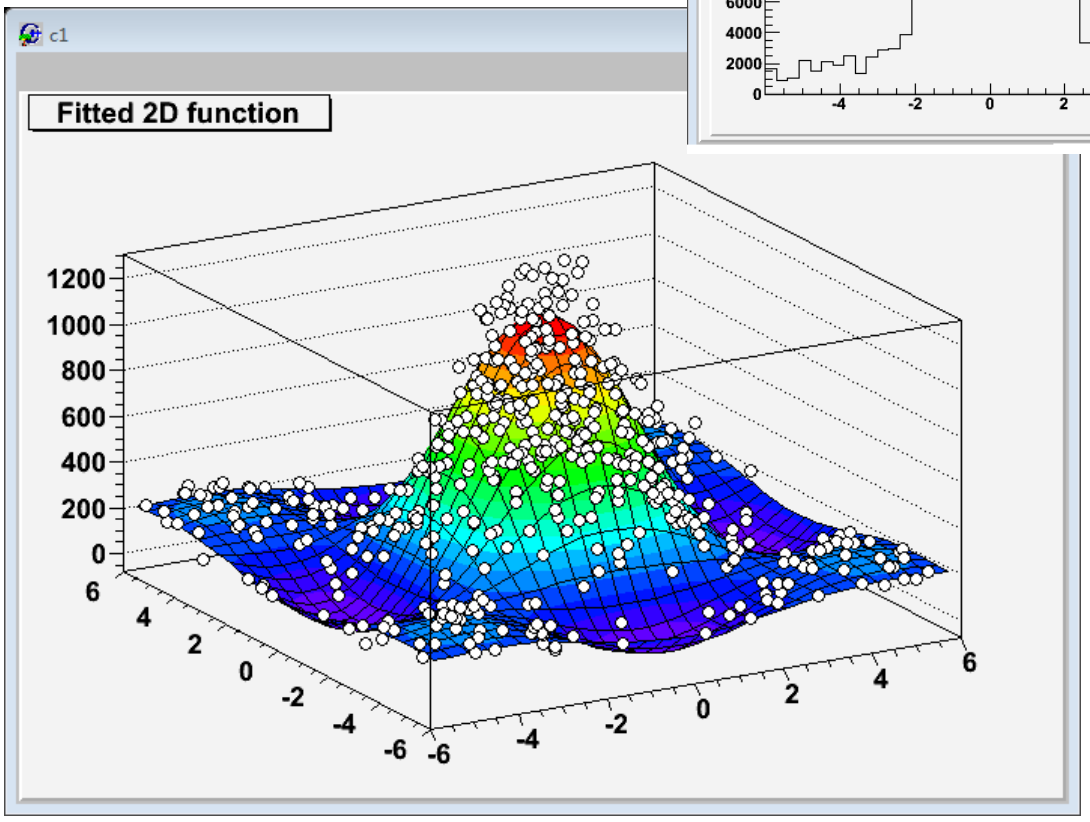
```
// Builds a polar graph in a square Canvas.

void macro3(){
     TCanvas* c = new TCanvas("myCanvas","myCanvas",600,600);
     double rmin=0;
     double rmax=TMath::Pi()*6;
     const int npoints=1000;
     Double_t r[npoints];
     Double_t theta[npoints];
     for (Int_t ipt = 0; ipt < npoints; ipt++) {
          r[ipt] = ipt*(rmax-rmin)/npoints+rmin;
          theta[ipt] = TMath::Sin(r[ipt]);
     }
     TGraphPolar grP1 (npoints,r,theta);
     grP1.SetTitle("A Fan");
     grP1.SetLineWidth(3);
     grP1.SetLineColor(2);
     grP1.DrawClone("AOL");
}
```



2D Graphs

macro4.C





siguang@pku.edu.cn

macro4.C

```
// Create, Draw and fit a TGraph2DErrors
void macro4(){
   gStyle->SetPalette(1);
   const double e = 0.3;
   const int nd = 500;

   TRandom3 my_random_generator;
   TF2 *f2 = new TF2("f2",
                     "1000*(([0]*sin(x)/x)*([1]*sin(y)/y))+200",
                     -6,6,-6,6);
   f2->SetParameters(1,1);
   TGraph2DErrors *dte = new TGraph2DErrors(nd);
   // Fill the 2D graph
   double rnd, x, y, z, ex, ey, ez;
   for (Int_t i=0; i<nd; i++) {
      f2->GetRandom2(x,y);
      // A random number in [-e,e]
      rnd = my_random_generator.Uniform(-e,e);
      z = f2->Eval(x,y)*(1+rnd);
      dte->SetPoint(i,x,y,z);
      ex = 0.05*my_random_generator.Uniform();
      ey = 0.05*my_random_generator.Uniform();
      ez = TMath::Abs(z*rnd);
      dte->SetPointError(i,ex,ey,ez);
   }
   // Fit function to generated data
   f2->SetParameters(0.7,1.5);  // set initial values for fit
   f2->SetTitle("Fitted 2D function");
   dte->Fit(f2);
```

macro4.C [continue]

```
    // Plot the result
    TCanvas *c1 = new TCanvas();
    f2->Draw("Surf1");
    dte->Draw("P0 Same");
    // Make the x and y projections
    TCanvas* c_p= new TCanvas("ProjCan",
                              "The Projections",1000,400);
    c_p->Divide(2,1);
    c_p->cd(1);
    dte->Project("x")->Draw();
    c_p->cd(2);
    dte->Project("y")->Draw();
}
```

Fit Options

```
void TH1::Fit(const char *fname, Option_t *option, Option_t *goption,
Axis_t xxmin, Axis_t xxmax)
```

- *option:The second parameter is the fitting option. Here is the list of fitting options:
- “W” Set all weights to 1 for non empty bins; ignore error bars
- “WW” Set all weights to 1 including empty bins; ignore error bars
- “I” Use integral of function in bin instead of value at bin center
- “L” Use log likelihood method (default is chi-square method)
- “U” Use a user specified fitting algorithm
- “Q” Quiet mode (minimum printing)
- “V” Verbose mode (default is between Q and V)
- “E” Perform better errors estimation using the Minos technique
- “M” Improve fit results
- “R” Use the range specified in the function range
- “N” Do not store the graphics function, do not draw
- “0” Do not plot the result of the fit. By default the fitted function is drawn unless the option “N” above is specified.

Fit Options

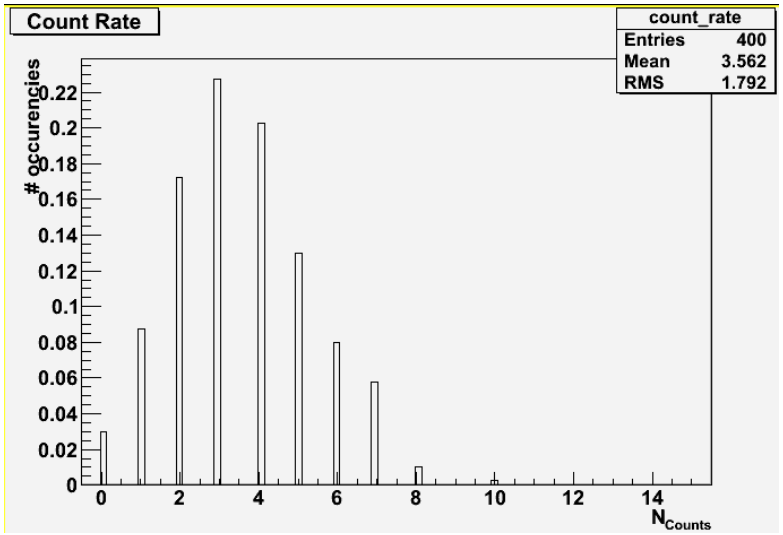
• “+” Add this new fitted function to the list of fitted functions (by default, the previous function is deleted and
only the last one is kept)
• “B”Use this option when you want to fix one or more parameters and the fitting function is like polN, expo,
landau, gaus.
• “LL”An improved Log Likelihood fit in case of very low statistics and when bincontentsare not integers. Do not
use this option if bin contents are large (greater than 100).
• “C”In case of linear fitting, don’t calculate the chisquare (saves time).
• “F”If fitting a polN, switch to Minuit fitter (by default, polN functions are fitted by the linear fitter).

• *goption:The third parameter is the graphics option that is the same as in the **TH1**::Draw (see the chapter Draw Options).
• xxmin, xxmax:Thee fourth and fifth parameters specify the range over which to apply the fit.

Your First Histogram

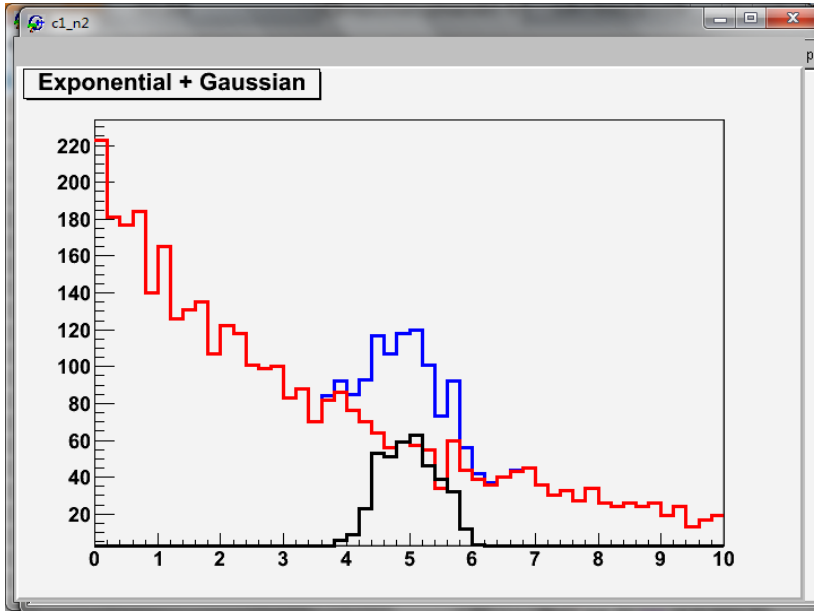
macro.C

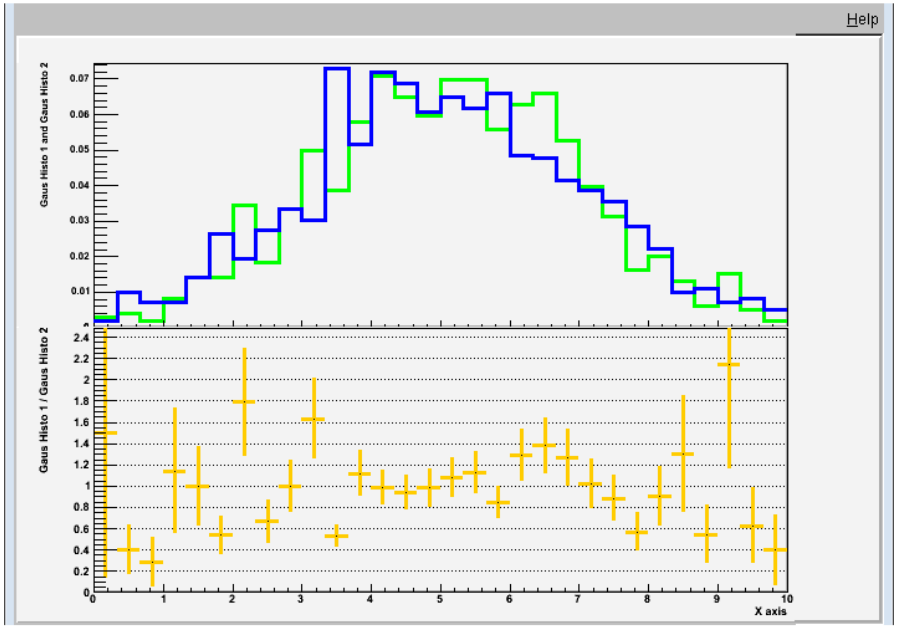
```
// Create, Fill and draw an Histogram which reproduces the
// counts of a scaler linked to a Geiger counter.

void macro5(){
    TH1F* cnt_r_h=new TH1F("count_rate",
                "Count Rate;N_{Counts};# occurencies",
                100, // Number of Bins
                -0.5, // Lower X Boundary
                15.5); // Upper X Boundary

    const float mean_count=3.6;
    TRandom3 rndgen;
    // simulate the measurements
    for (int imeas=0;imeas<400;imeas++)
        cnt_r_h->Fill(rndgen.Poisson(mean_count));

    TCanvas* c= new TCanvas();
    cnt_r_h->Draw();

    TCanvas* c_norm= new TCanvas();
    cnt_r_h->DrawNormalized();

    // Print summary
    cout << "Moments of Distribution:\n"
         << " - Mean = " << cnt_r_h->GetMean() << " +- "
                         << cnt_r_h->GetMeanError() << "\n"
         << " - RMS = " << cnt_r_h->GetRMS() << " +- "
                        << cnt_r_h->GetRMSError() << "\n"
         << " - Skewness = " << cnt_r_h->GetSkewness() << "\n"
         << " - Kurtosis = " << cnt_r_h->GetKurtosis() << "\n";
}
```



Add and Divide Histograms

macro6.C





右上图的y轴可用SetRangeUser(0.001,0.08)改进
消除露出一半的数字

siguang@pku.edu.cn

macro6.C

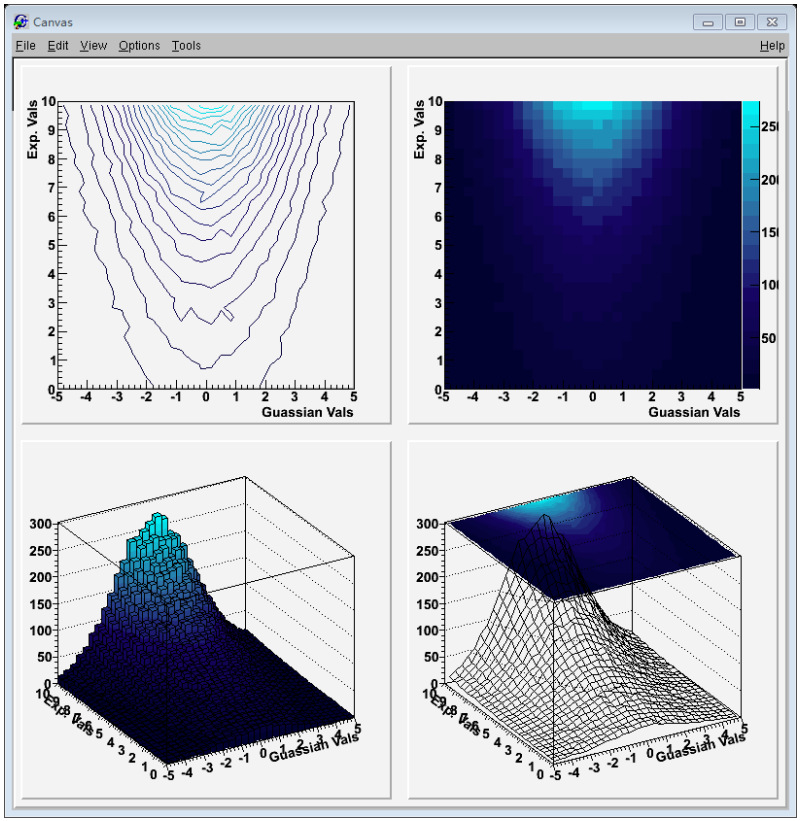
```
// Divide and add 1D Histograms

void format_h(TH1F* h, int linecolor){
    h->SetLineWidth(3);
    h->SetLineColor(linecolor);
    }

void macro6(){

    TH1F* sig_h=new TH1F("sig_h","Signal Histo",50,0,10);
    TH1F* gaus_h1=new TH1F("gaus_h1","Gauss Histo 1",30,0,10);
    TH1F* gaus_h2=new TH1F("gaus_h2","Gauss Histo 2",30,0,10);
    TH1F* bkg_h=new TH1F("exp_h","Exponential Histo",50,0,10);

    // simulate the measurements
    TRandom3 rndgen;
    for (int imeas=0;imeas<4000;imeas++){
        exp_h->Fill(rndgen.Exp(4));
        if (imeas%4==0) gaus_h1->Fill(rndgen.Gaus(5,2));
        if (imeas%4==0) gaus_h2->Fill(rndgen.Gaus(5,2));
        if (imeas%10==0)sig_h->Fill(rndgen.Gaus(5,.5));}

    // Format Histograms
    TH1F* histos[4]={sig_h,bkg_h,gaus_h1,gaus_h2};
    for (int i=0;i<4;++i){
        histos[i]->Sumw2(); // *Very* Important
        format_h(histos[i],i+1);
        }
```

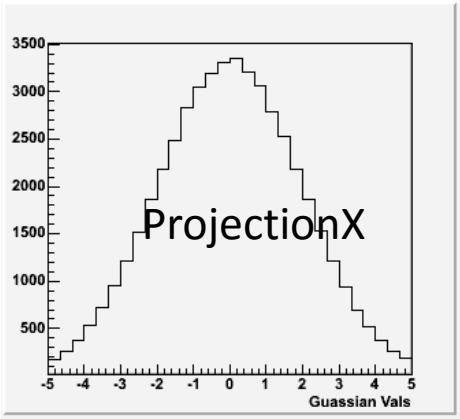
macro6.C

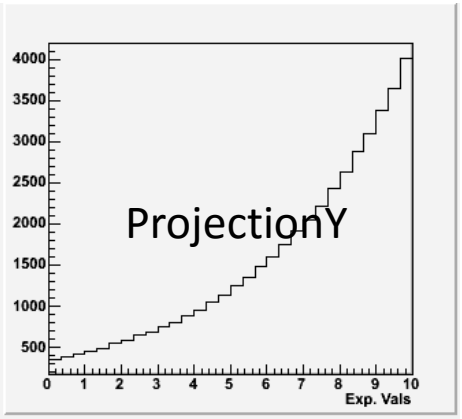
```
// Sum
TH1F* sum_h= new TH1F(*bkg_h);
sum_h->Add(sig_h,1.);
sum_h->SetTitle("Exponential + Gaussian");
format_h(sum_h,kBlue);

TCanvas* c_sum= new TCanvas();
sum_h->Draw("hist");
bkg_h->Draw("SameHist");
sig_h->Draw("SameHist");

// Divide
TH1F* dividend=new TH1F(*gaus_h1);
dividend->Divide(gaus_h2);

// Graphical Maquillage
dividend->SetTitle(";X axis;Gaus Histo 1 / Gaus Histo 2");
format_h(dividend,kOrange);
gaus_h1->SetTitle(";;Gaus Histo 1 and Gaus Histo 2");
gStyle->SetOptStat(0);

TCanvas* c_divide= new TCanvas();
c_divide->Divide(1,2,0,0);
c_divide->cd(1);
c_divide->GetPad(1)->SetRightMargin(.01);
gaus_h1->DrawNormalized("Hist");
gaus_h2->DrawNormalized("HistSame");
c_divide->cd(2);
dividend->GetYaxis()->SetRangeUser(0,2.49);
c_divide->GetPad(2)->SetGridy();
c_divide->GetPad(2)->SetRightMargin(.01);
dividend->Draw();
}
```

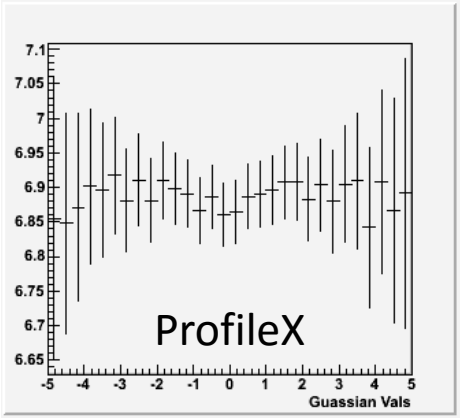
Two-dimensional Histograms

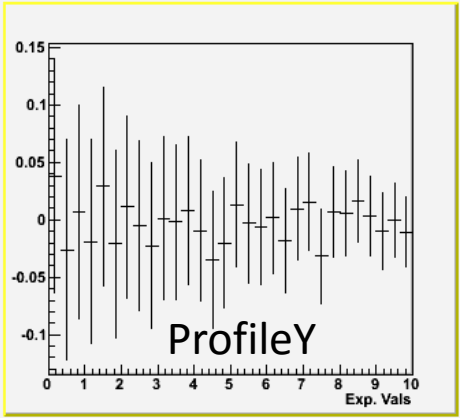
Output of macro7.C











siguang@pku.edu.cn

macro7.C

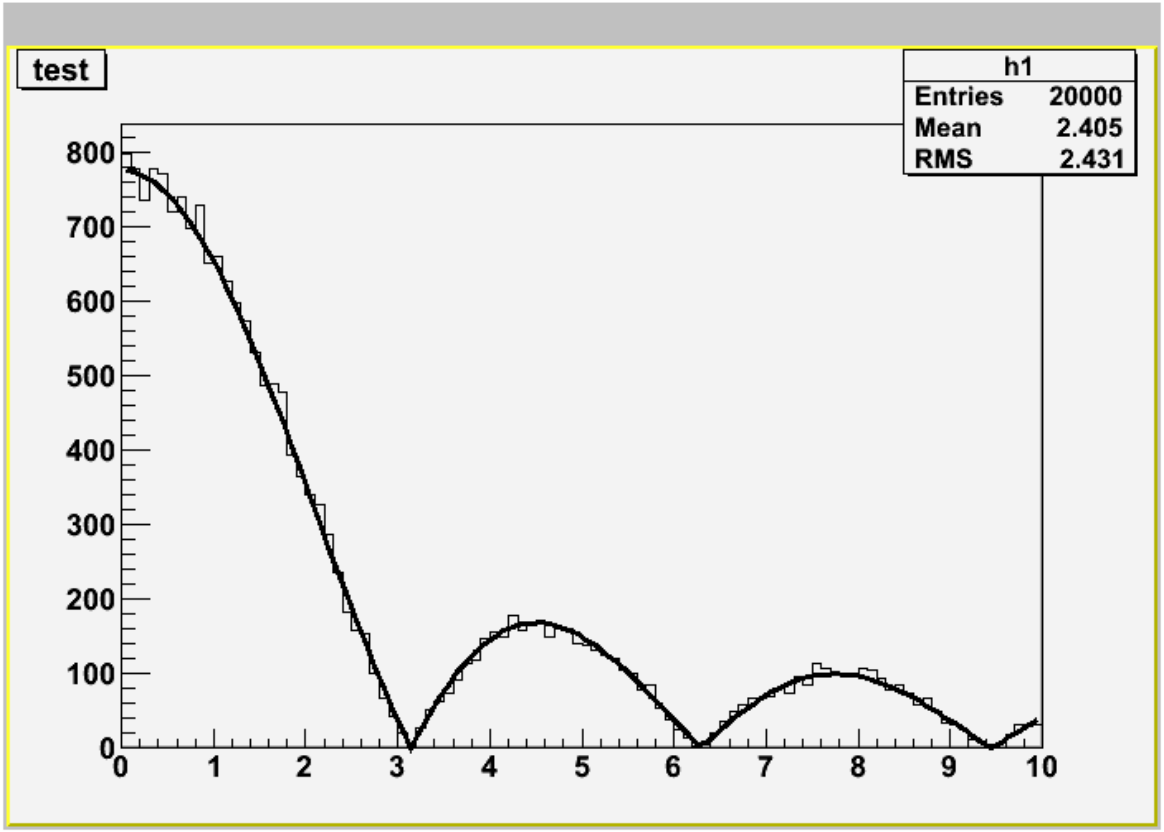
```
// Draw a Bidimensional Histogram in many ways
// together with its profiles and projections

void macro7(){
    gStyle->SetPalette(53);
    gStyle->SetOptStat(0);
    gStyle->SetOptTitle(0);

    TH2F bidi_h("bidi_h","2D Histo;Guassian Vals;Exp. Vals",
                30,-5,5,  // X axis
                30,0,10); // Y axis

    TRandom3 rgen;
    for (int i=0;i<500000;i++)
        bidi_h.Fill(rgen.Gaus(0,2),10-rgen.Exp(4),.1);

    TCanvas* c=new TCanvas("Canvas","Canvas",800,800);
    c->Divide(2,2);
    c->cd(1);bidi_h.DrawClone("Cont1");
    c->cd(2);bidi_h.DrawClone("Colz");
    c->cd(3);bidi_h.DrawClone("lego2");
    c->cd(4);bidi_h.DrawClone("surf3");

    // Profiles and Projections
    TCanvas* c2=new TCanvas("Canvas2","Canvas2",800,800);
    c2->Divide(2,2);
    c2->cd(1);bidi_h.ProjectionX()->DrawClone();
    c2->cd(2);bidi_h.ProjectionY()->DrawClone();
    c2->cd(3);bidi_h.ProfileX()->DrawClone();
    c2->cd(4);bidi_h.ProfileY()->DrawClone();
}
```

自定义函数 (Cmyfit.C)

```
#include <TF1.h>
#include <TH1F.h>
#include <TROOT.h>

// Macro myfunc.C
   Double_t myfunction(Double_t *x, Double_t *par)
   {
      Float_t xx =x[0];
      Double_t f = TMath::Abs(par[0]*sin(par[1]*xx)/xx);
      return f;
   }
   void myfunc()
   {
      TF1 *f1 = new TF1("myfunc",myfunction,0,10,2);
      f1->SetParameters(2,1);
      f1->SetParNames("constant","coefficient");
      f1->Draw();
   }
   void myfit()
   {
      TH1F *h1=new TH1F("h1","test",100,0,10);
      h1->FillRandom("myfunc",20000);
      TF1 *f1=(TF1 *)gROOT->GetFunction("myfunc");
      f1->SetParameters(800,1);
      h1->Fit("myfunc");
   }
```

run

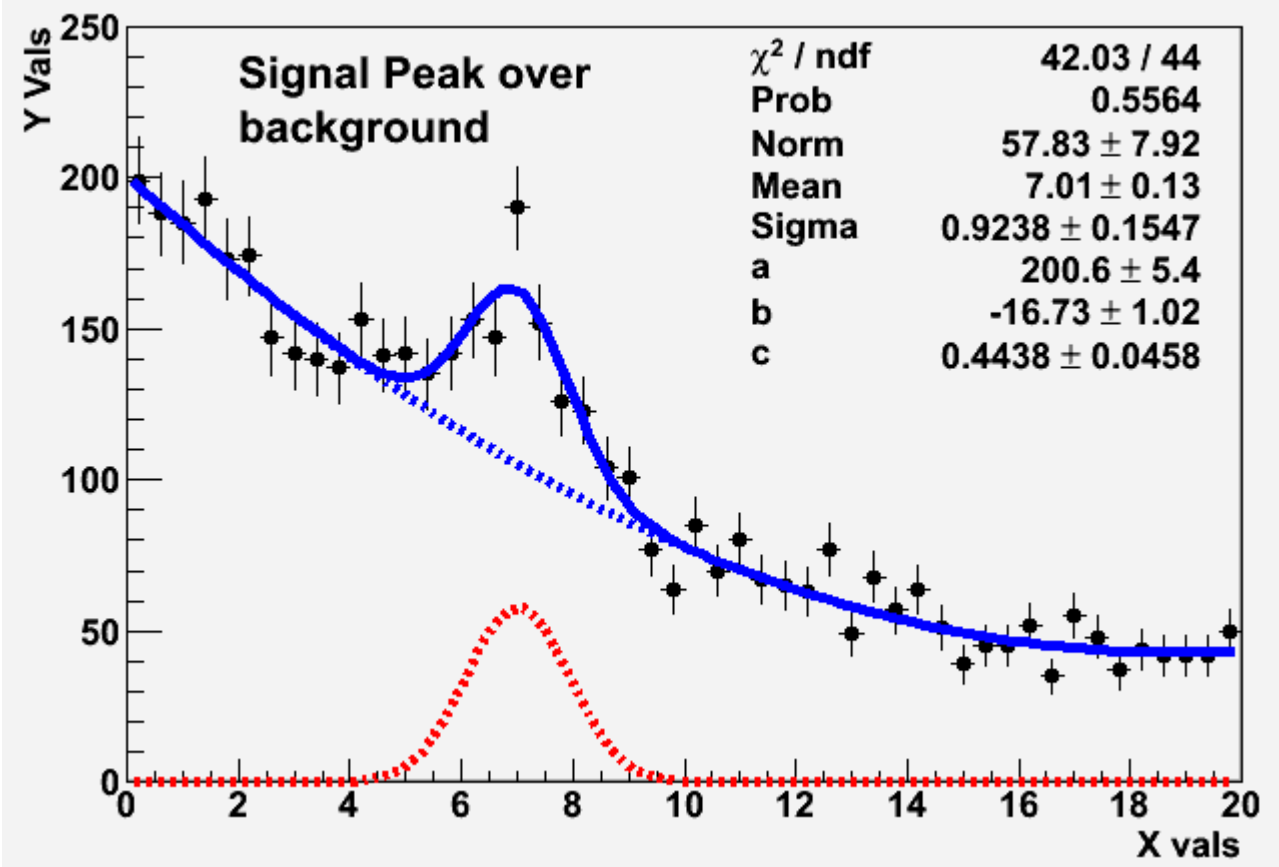
root [1].L Cmyfit.C
root [1] myfunc()
root [2] myfit()



siguang@pku.edu.cn

Functions and Parameter Estimation

Output of macro8.C



siguang@pku.edu.cn

macro8.C

```
void format_line(TAttLine* line,int col,int sty){
    line->SetLineWidth(5); line->SetLineColor(col);
    line->SetLineStyle(sty);}

double the_gausppar(double* vars, double* pars){
    return pars[0]*TMath::Gaus(vars[0],pars[1],pars[2])+
        pars[3]+pars[4]*vars[0]+pars[5]*vars[0]*vars[0];}

int macro8(){
    gStyle->SetOptTitle(0); gStyle->SetOptStat(0);
    gStyle->SetOptFit(1111); gStyle->SetStatBorderSize(0);
    gStyle->SetStatX(.89); gStyle->SetStatY(.89);

    TF1 parabola("parabola","[0]+[1]*x+[2]*x**2",0,20);
    format_line(&parabola,kBlue,2);

    TF1 gaussian("gaussian","[0]*TMath::Gaus(x,[1],[2])",0,20);
    format_line(&gaussian,kRed,2);
```

macro8.C [continue]

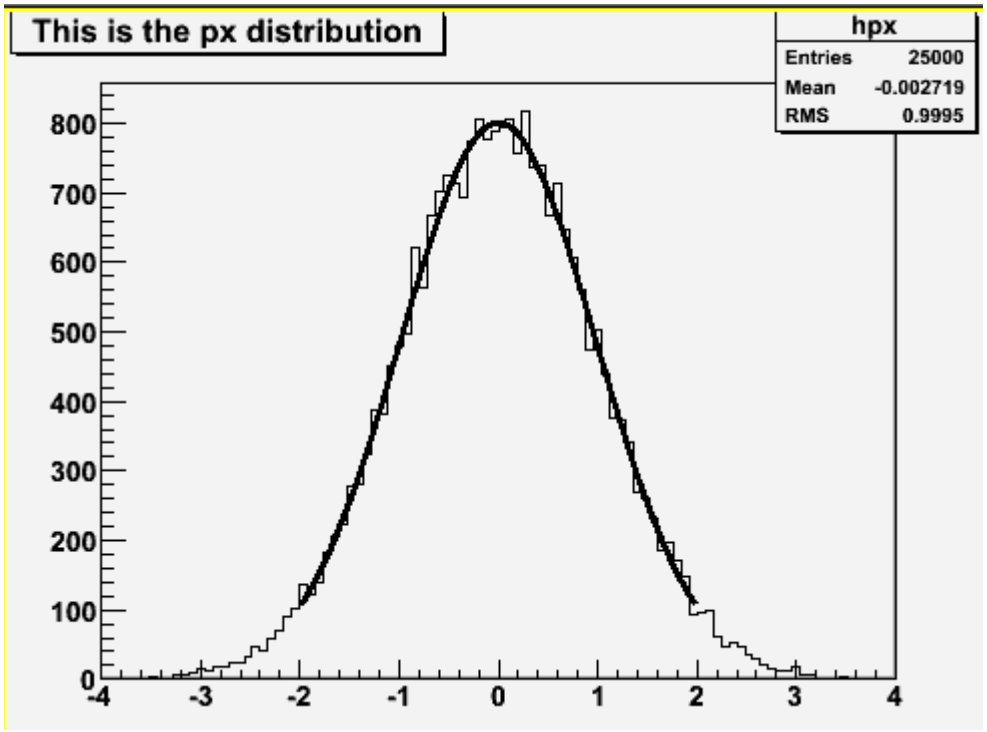
```
TF1 gausppar("gausppar",the_gausppar,-0,20,6);
double a=15; double b=-1.2; double c=.03;
double norm=4; double mean=7; double sigma=1;
gausppar.SetParameters(norm,mean,sigma,a,b,c);
gausppar.SetParNames("Norm","Mean","Sigma","a","b","c");
format_line(&gausppar,kBlue,1);

TH1F histo("histo","Signal plus background;X vals;Y Vals",
           50,0,20);
histo.SetMarkerStyle(8);

// Fake the data
for (int i=1;i<=5000;++i) histo.Fill(gausppar.GetRandom());

// Reset the parameters before the fit and set
// by eye a peak at 6 with an area of more or less 50
gausppar.SetParameter(0,50);
gausppar.SetParameter(1,6);
int npar=gausppar.GetNpar();
for (int ipar=2;ipar<npar;++ipar)
    gausppar.SetParameter(ipar,1);

// perform fit ...
TFitResultPtr frp = histo.Fit(&gausppar, "S");

// ... and retrieve fit results
frp->Print(); // print fit results
// get covariance Matrix an print it
TMatrixDSym covMatrix (frp->GetCovarianceMatrix());
covMatrix.Print();
```

macro8.C [continue]

```
    // Set the values of the gaussian and parabola
    for (int ipar=0;ipar<3;ipar++){
        gaussian.SetParameter(ipar,
                              gausppar.GetParameter(ipar));
        parabola.SetParameter(ipar,
                              gausppar.GetParameter(ipar+3));}

    histo.GetYaxis()->SetRangeUser(0,250);
    histo.DrawClone("PE");
    parabola.DrawClone("Same"); gaussian.DrawClone("Same");
    TLatex latex(2,220,
                 "#splitline{Signal Peak over}{background}");
    latex.DrawClone("Same");
}
```

用户自定义函数

$ROOTSYS/tutorials/fit/myfit.C



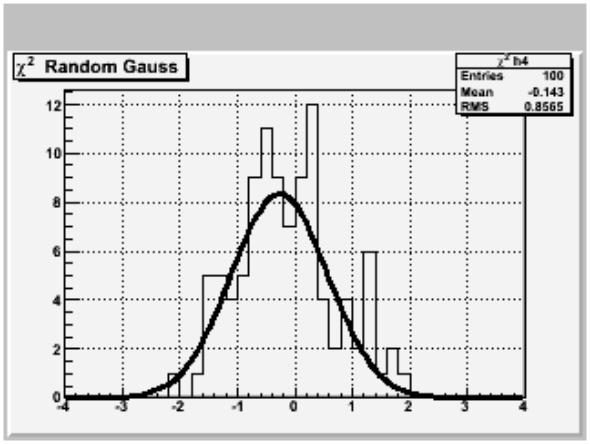
myfit.C

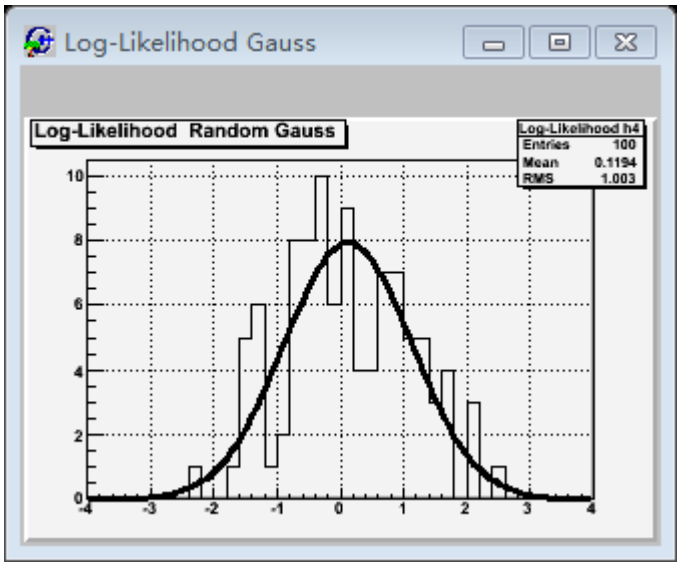
```
// Get in memory an histogram from a root file and fit a user defined function.
// Note that a user defined function must always be defined
// as in this example:
//  - first parameter: array of variables (in this example only 1-dimension)
//  - second parameter: array of parameters
// Note also that in case of user defined functions, one must set
// an initial value for each parameter.
//Author: Rene Brun

Double_t fitf(Double_t *x, Double_t *par)
{
   Double_t arg = 0;
   if (par[2] != 0) arg = (x[0] - par[1])/par[2];

   Double_t fitval = par[0]*TMath::Exp(-0.5*arg*arg);
   return fitval;
}
void myfit()
{
   TString dir = gSystem->UnixPathName(gInterpreter->GetCurrentMacroName());
   dir.ReplaceAll("myfit.C","../hsimple.C");
   dir.ReplaceAll("/./","/");
   if (!gInterpreter->IsLoaded(dir.Data())) gInterpreter->LoadMacro(dir.Data());

   TFile *hsimple = (TFile*)gROOT->ProcessLineFast("hsimple(1)");
   if (!hsimple) return;

   TCanvas *c1 = new TCanvas("c1","the fit canvas",500,400);

   TH1F *hpx = (TH1F*)hsimple->Get("hpx");
```

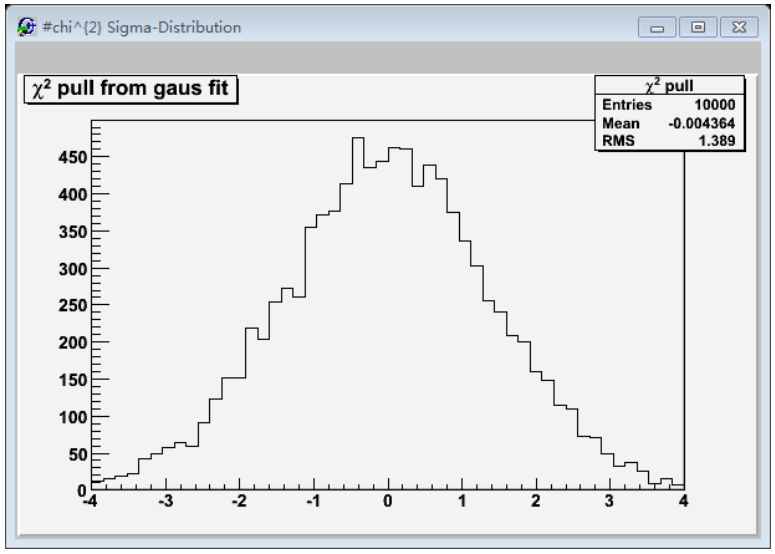
myfit.C [continue]

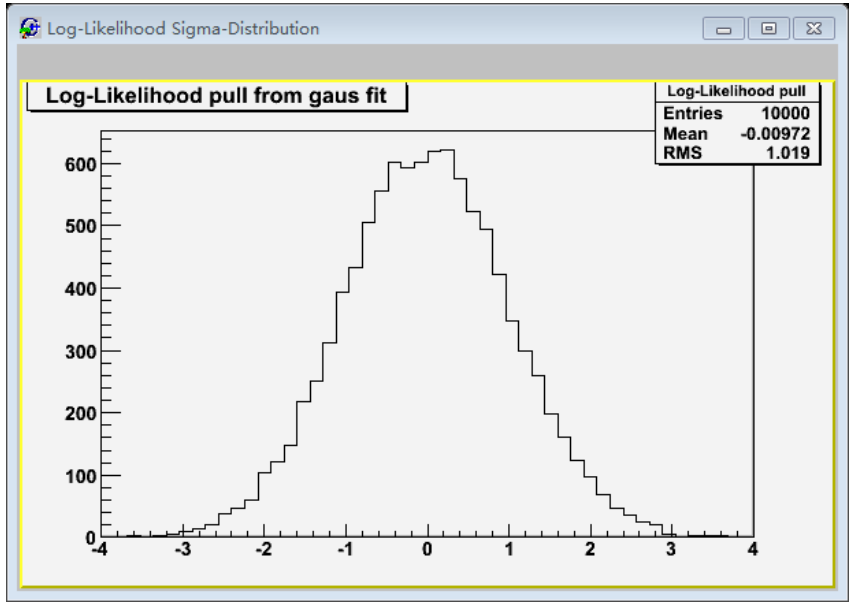
```
// Creates a Root function based on function fitf above
   TF1 *func = new TF1("fitf",fitf,-2,2,3);

// Sets initial values and parameter names
   func->SetParameters(100,0,1);
   func->SetParNames("Constant","Mean_value","Sigma");

// Fit histogram in range defined by function
   hpx->Fit(func,"r");

// Gets integral of function between fit limits
   printf("Integral of function = %g\n",func->Integral(-2,2));
}
```

Output of macro9.C









siguang@pku.edu.cn

macro9.C

```
// Toy Monte Carlo example.
// Check pull distribution to compare chi2 and binned
// log-likelihood methods.

pull( int n_toys = 10000,
      int n_tot_entries = 100,
      int nbins = 40,
      bool do_chi2=true ){

  TString method_prefix("Log-Likelihood ");
  if (do_chi2)
    method_prefix="#chi^{2} ";

  // Create histo
  TH1F* h4 = new TH1F(method_prefix+"h4",
                      method_prefix+" Random Gauss",
                      nbins,-4,4);
  h4->SetMarkerStyle(21);
  h4->SetMarkerSize(0.8);
  h4->SetMarkerColor(kRed);

  // Histogram for sigma and pull
  TH1F* sigma = new TH1F(method_prefix+"sigma",
                         method_prefix+"sigma from gaus fit",
                         50,0.5,1.5);
  TH1F* pull = new TH1F(method_prefix+"pull",
                        method_prefix+"pull from gaus fit",
                        50,-4.,4.);

  // Make nice canvases
  TCanvas* c0 = new TCanvas(method_prefix+"Gauss",
                            method_prefix+"Gauss",0,0,320,240);
  c0->SetGrid();
```


siguang@pku.edu.cn

macro9.C [continue]

```
// Make nice canvases
TCanvas* c1 = new TCanvas(method_prefix+"Result",
                          method_prefix+"Sigma-Distribution",
                          0,300,600,400);
c0->cd();

float sig, mean;
for (int i=0; i<n_toys; i++){
  // Reset histo contents
  h4->Reset();
  // Fill histo
  for ( int j = 0; j<n_tot_entries; j++ )
    h4->Fill(gRandom->Gaus());
  // perform fit
  if (do_chi2) h4->Fit("gaus","q"); // Chi2 fit
  else h4->Fit("gaus","lq"); // Likelihood fit
  // some control output on the way
  if (!(i%100)){
    h4->Draw("ep");
    c0->Update();}

  // Get sigma from fit
  TF1 *fit = h4->GetFunction("gaus");
  sig = fit->GetParameter(2);
  mean= fit->GetParameter(1);
  sigma->Fill(sig);
  pull->Fill(mean/sig * sqrt(n_tot_entries));
} // end of toy MC loop
```

macro9.C [continue]

```
  // print result
  c1->cd();
  pull->Draw();
}

void macro9(){
  int n_toys=10000;
  int n_tot_entries=100;
  int n_bins=40;
  cout << "Performing Pull Experiment with chi2 \n";
  pull(n_toys,n_tot_entries,n_bins,true);
  cout << "Performing Pull Experiment with Log Likelihood\n";
  pull(n_toys,n_tot_entries,n_bins,false);
}
```

怎样编译运行？试一试

As a very simple yet powerful quantity to check the quality of the fit results, we construct for each pseudo-data set the so-called "pull", the difference of the estimated and the true value of a parameter, normalised to the estimated error on the parameter, $\frac{(p_{estim}-p_{true})}{\sigma_p}$. If everything is OK, the distribution of the pull values is a standard normal distribution, i.e. a Gaussian distribution centred around zero with a standard deviation of one.

Storing ROOT Objects

```
void write_to_file(){

    // Instance of our histogram
    TH1F h("my_histogram","My Title;X;# of entries",100,-5,5);

    // Let's fill it randomly
    h.FillRandom("gaus");

    // Let's open a TFile
    TFile out_file("my_rootfile.root","RECREATE");

    // Write the histogram in the file
    h.Write();

    // Close the file
    out_file.Close();
}
```

read_from_file.C

```
void read_from_file(){

    // Let's open the TFile
    TFile* in_file= new TFile("my_rootfile.root");

    // Get the Histogram out
    //  TH1F* h = (TH1F*) in_file->GetObjectChecked("my_histogram","TH1F");

    TH1F* h = (TH1F*) in_file->Get("my_histogram");

    // Draw it
    h->Draw();
}
```

TMVA - Toolkit for Multivariate Data Analysis

534/tmva/test

- To look the README
  - *emacs README &*

How to run the example

- First, TMVAClassification.C needs to be run to perform the training. You can edit TMVAClassification.C (boolean flags at the head of the file) to enable (disable) the methods you would (noTMVAClassification.C needs to be run to perform the trainingt) like to use. Then run
  - *> root TMVAClassification.C*
- at the end a file TMVA.root is written, and a GUI opens. Once TMVA.root exists, the GUI can also be started via
  - *> root TMVAGui.C*
- You can also run
  - >root TMVAClassificationApplication.C

  which evaluates the methods on the signal data from the toy tmva_example.root and writes the result to TMVApp.root. However, TMVAClassificationApplication is more of a pedagogical example for the usage of TMVA inside your own analysis framework.

TMVAClassification.C

TMVAClassification.C

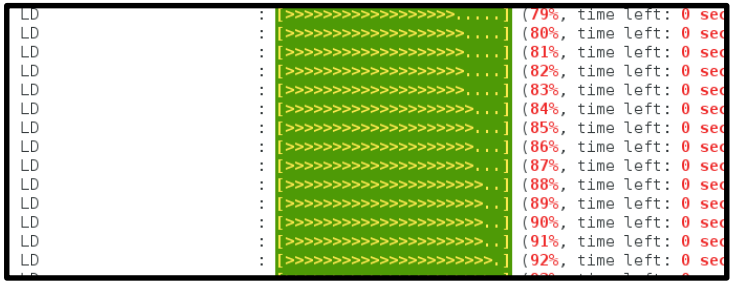
```
void TMVAClassification( TString myMethodList = "" )
{
   // The explicit loading of the shared libTMVA is done in TMVAlogon.C, defined in .rootrc
   // if you use your private .rootrc, or run from a different directory, please copy the
   // corresponding lines from .rootrc

   // methods to be processed can be given as an argument; use format:
   //
   // mylinux~> root -l TMVAClassification.C\(\"myMethod1,myMethod2,myMethod3\"\)
   //
   // if you like to use a method via the plugin mechanism, we recommend using
   //
   // mylinux~> root -l TMVAClassification.C\(\"P_myMethod\"\)
   // (an example is given for using the BDT as plugin (see below),
   // but of course the real application is when you write your own
   // method based)
```

TMVAClassification.C [continue]

```
//---------------------------------------------------------------
// This loads the library
TMVA::Tools::Instance();

// to get access to the GUI and all tmva macros
TString thisdir = gSystem->DirName(gInterpreter->GetCurrentMacroName());
gROOT->SetMacroPath(thisdir + ":" + gROOT->GetMacroPath());
gROOT->ProcessLine(".L TMVAGui.C");

// Default MVA methods to be trained + tested
std::map<std::string,int> Use;

// --- Cut optimisation
Use["Cuts"]            = 1;
Use["CutsD"]           = 1;
Use["CutsPCA"]         = 0;
Use["CutsGA"]          = 0;
Use["CutsSA"]          = 0;
//
// --- 1-dimensional likelihood ("naive Bayes estimator")
Use["Likelihood"]      = 1;
Use["LikelihoodD"]     = 0; // the "D" extension indicates decorrelated input variables (see option strings)
```

TMVAClassification.C [continue]

```
// Create a ROOT output file where TMVA will store ntuples, histograms, etc.
TString outfileName( "TMVA.root" );
TFile* outputFile = TFile::Open( outfileName, "RECREATE" );

// Create the factory object. Later you can choose the methods
// whose performance you'd like to investigate. The factory is
// the only TMVA object you have to interact with
//
// The first argument is the base of the name of all the
// weightfiles in the directory weight/
//
// The second argument is the output file for the training results
// All TMVA output can be suppressed by removing the "!" (not) in
// front of the "Silent" argument in the option string
TMVA::Factory *factory = new TMVA::Factory( "TMVAClassification", outputFile,
                                            "!V:!Silent:Color:DrawProgressBar:Transformations=I;D;P;G,D:AnalysisType=Classificati
on" );
```



"V", "Verbose flag";
"Color", "Flag for coloured screen output (default: True, if in batch mode: False)"
"Transformations", "List of transformations to test; formatting example: "Transformations=I;D;P;G,D", for identity, decorrelation, PCA, and Gaussianisation followed by decorrelation transformations"
"Silent", "Batch mode: boolean silent flag inhibiting any output from TMVA after the creation of the factory class object (default: False)“
"DrawProgressBar", "Draw progress bar to display training, testing and evaluation schedule (default: True)");

TMVAClassification.C [continue]

```
// Define the input variables that shall be used for the MVA training
// note that you may also use variable expressions, such as: "3*var1/var2*abs(var3)"
// [all types of expressions that can also be parsed by TTree::Draw( "expression" )]
factory->AddVariable( "myvar1 := var1+var2", 'F' );
factory->AddVariable( "myvar2 := var1-var2", "Expression 2", "", 'F' );
factory->AddVariable( "var3",                "Variable 3", "units", 'F' );
factory->AddVariable( "var4",                "Variable 4", "units", 'F' );

// You can add so-called "Spectator variables", which are not used in the MVA training,
// but will appear in the final "TestTree" produced by TMVA. This TestTree will contain the
// input variables, the response values of all trained MVAs, and the spectator variables
factory->AddSpectator( "spec1 := var1*2",  "Spectator 1", "units", 'F' );
factory->AddSpectator( "spec2 := var1*3",  "Spectator 2", "units", 'F' );

// Read training and test data
// (it is also possible to use ASCII format as input -> see TMVA Users Guide)
TString fname = "./tmva_class_example.root";

if (gSystem->AccessPathName( fname ))  // file does not exist in local directory
   gSystem->Exec("curl -O http://root.cern.ch/files/tmva_class_example.root");
```

F for Float

TMVAClassification.C [continue]

```
TFile *input = TFile::Open( fname );

std::cout << "--- TMVAClassification       : Using input file: " << input->GetName() << std::endl;

// --- Register the training and test trees

TTree *signal     = (TTree*)input->Get("TreeS");
TTree *background = (TTree*)input->Get("TreeB");

// global event weights per tree (see below for setting event-wise weights)
Double_t signalWeight     = 1.0;
Double_t backgroundWeight = 1.0;

// You can add an arbitrary number of signal or background trees
factory->AddSignalTree    ( signal,     signalWeight     );
factory->AddBackgroundTree( background, backgroundWeight );
```

TMVAClassification.C [continue]

```
// To give different trees for training and testing, do as follows:
//    factory->AddSignalTree( signalTrainingTree, signalTrainWeight, "Training" );
//    factory->AddSignalTree( signalTestTree,     signalTestWeight,  "Test" );

// Use the following code instead of the above two or four lines to add signal and background
// training and test events "by hand"
// NOTE that in this case one should not give expressions (such as "var1+var2") in the input
//      variable definition, but simply compute the expression before adding the event
//
//     // --- begin ----------------------------------------------------------
//     std::vector<Double_t> vars( 4 ); // vector has size of number of input variables
//     Float_t  treevars[4], weight;
//
//     // Signal
//     for (UInt_t ivar=0; ivar<4; ivar++) signal->SetBranchAddress( Form( "var%i", ivar+1 ), &(treevars[ivar]) );
//     for (UInt_t i=0; i<signal->GetEntries(); i++) {
//        signal->GetEntry(i);
//        for (UInt_t ivar=0; ivar<4; ivar++) vars[ivar] = treevars[ivar];
//        // add training and test events; here: first half is training, second is testing
//        // note that the weight can also be event-wise
//        if (i < signal->GetEntries()/2.0) factory->AddSignalTrainingEvent( vars, signalWeight );
//        else                              factory->AddSignalTestEvent    ( vars, signalWeight );
//     }
//
```

TMVAClassification.C [continue]

```
//     // Background (has event weights)
//     background->SetBranchAddress( "weight", &weight );
//     for (UInt_t ivar=0; ivar<4; ivar++) background->SetBranchAddress( Form( "var%i", ivar+1 ), &(treevars[ivar]) );
//     for (UInt_t i=0; i<background->GetEntries(); i++) {
//        background->GetEntry(i);
//        for (UInt_t ivar=0; ivar<4; ivar++) vars[ivar] = treevars[ivar];
//        // add training and test events; here: first half is training, second is testing
//        // note that the weight can also be event-wise
//        if (i < background->GetEntries()/2) factory->AddBackgroundTrainingEvent( vars, backgroundWeight*weight );
//        else                                factory->AddBackgroundTestEvent    ( vars, backgroundWeight*weight );
//     }
      // --- end ------------------------------------------------------------
//
// --- end of tree registration
```

TMVAClassification.C [continue]

```
// Tell the factory how to use the training and testing events
//
// If no numbers of events are given, half of the events in the tree are used
// for training, and the other half for testing:
//    factory->PrepareTrainingAndTestTree( mycut, "SplitMode=random:!V" );
// To also specify the number of testing events, use:
//    factory->PrepareTrainingAndTestTree( mycut,
//                                         "NSigTrain=3000:NBkgTrain=3000:NSigTest=3000:NBkgTest=3000:SplitMode=Random:!V" );
factory->PrepareTrainingAndTestTree( mycuts, mycutb,
                                     "nTrain_Signal=0:nTrain_Background=0:SplitMode=Random:NormMode=NumEvents:!V" );
```

```
void TMVA::Factory::PrepareTrainingAndTestTree( TCut sigcut, TCut bkgcut, const TString& splitOpt )
{
   // prepare the training and test trees

   // if event-wise data assignment, add local trees to dataset first
   SetInputTreesFromEventAssignTrees();

   Log() << kINFO << "Preparing trees for training and testing..." << Endl;
   AddCut( sigcut, "Signal"  );
   AddCut( bkgcut, "Background" );

   DefaultDataSetInfo().SetSplitOptions( splitOpt );
}
```

TMVAClassification.C [continue]

```
// ---- Book MVA methods
//
// Please lookup the various method configuration options in the corresponding cxx files, eg:
// src/MethoCuts.cxx, etc, or here: http://tmva.sourceforge.net/optionRef.html
// it is possible to preset ranges in the option string in which the cut optimisation should be done:
// "...:CutRangeMin[2]=-1:CutRangeMax[2]=1"...", where [2] is the third input variable

// Cut optimisation
if (Use["Cuts"])
   factory->BookMethod( TMVA::Types::kCuts, "Cuts",
                        "!H:!V:FitMethod=MC:EffSel:SampleSize=200000:VarProp=FSmart" );

if (Use["CutsD"])
   factory->BookMethod( TMVA::Types::kCuts, "CutsD",
                        "!H:!V:FitMethod=MC:EffSel:SampleSize=200000:VarProp=FSmart:VarTransform=Decorrelate" );

if (Use["CutsPCA"])
   factory->BookMethod( TMVA::Types::kCuts, "CutsPCA",
                        "!H:!V:FitMethod=MC:EffSel:SampleSize=200000:VarProp=FSmart:VarTransform=PCA" );
```

TMVAClassification.C [continue]

http://tmva.sourceforge.net/optionRef.html

```
   // TMVA ANN: MLP (recommended ANN) -- all ANNs in TMVA are Multilayer Perceptrons
   if (Use["MLP"])
      factory->BookMethod( TMVA::Types::kMLP, "MLP", "H:!V:NeuronType=tanh:VarTransform=N:NCycles=600:HiddenLayers=N+5:TestRate=5:!U
seRegulator" );
```

Configuration options reference for MVA method: *MLP*

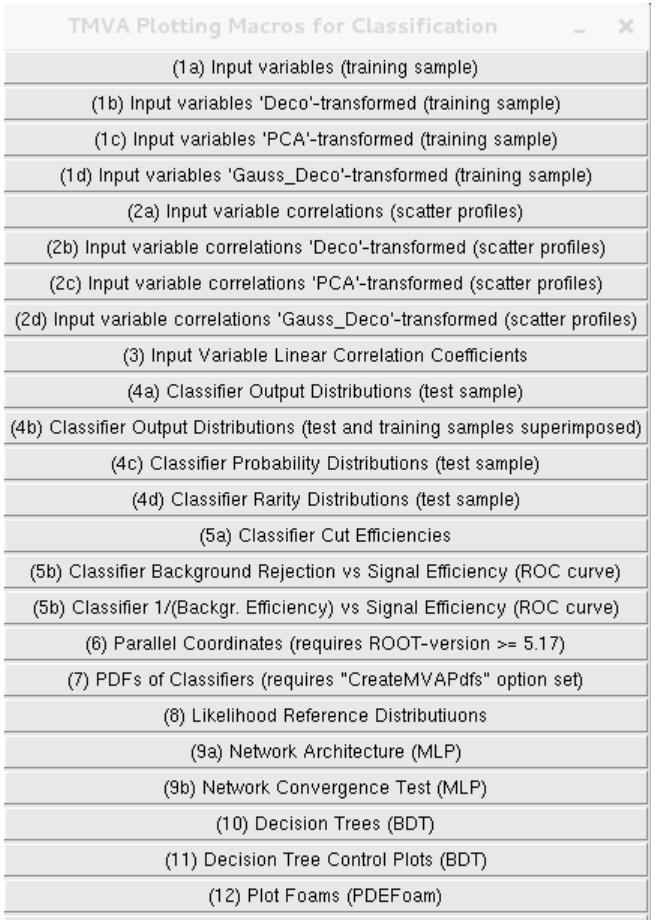
Option	Array	Default value	Predefined values	Description
NCycles	No	500	–	Number of training cycles
HiddenLayers	No	N,N-1	–	Specification of hidden layer architecture
NeuronType	No	sigmoid	–	Neuron activation function type
RandomSeed	No	1	–	Random seed for initial synapse weights (0 means unique seed for each run; default value '1')
EstimatorType	No	MSE	MSE, CE, linear, sigmoid, tanh, radial	MSE (Mean Square Estimator) for Gaussian Likelihood or CE(Cross-Entropy) for Bernoulli Likelihood
TestRate	No	10	–	Test for overtraining performed at each #th epochs
V	No	False	–	Verbose output (short form of VerbosityLevel below - overrides the latter one)
VerbosityLevel	No	Default	Default, Debug, Verbose, Info, Warning, Error, Fatal	Verbosity level
VarTransform	No	None	–	List of variable transformations performed before training, e.g., D_Background,P_Signal,G,N_AllClasses for: Decorrelation, PCA-transformation, Gaussianisation, Normalisation, each for the given class of events ('AllClasses' denotes all events of all classes, if no class indication is given, 'All' is assumed)
H	No	False	–	Print method-specific help message
UseRegulator	No	False	–	Use regulator to avoid over-training

TMVAClassification.C [continue]

```
   // ---- Now you can tell the factory to train, test, and evaluate the MVAs

   // Train MVAs using the set of training events
   factory->TrainAllMethods();

   // ---- Evaluate all MVAs using the set of test events
   factory->TestAllMethods();

   // ----- Evaluate and compare performance of all configured MVAs
   factory->EvaluateAllMethods();

   // --------------------------------------------------------------

   // Save the output
   outputFile->Close();

   std::cout << "==> Wrote root file: " << outputFile->GetName() << std::endl;
   std::cout << "==> TMVAClassification is done!" << std::endl;

   delete factory;
   // Launch the GUI for the root macros
   if (!gROOT->IsBatch()) TMVAGui( outfileName );
}
```

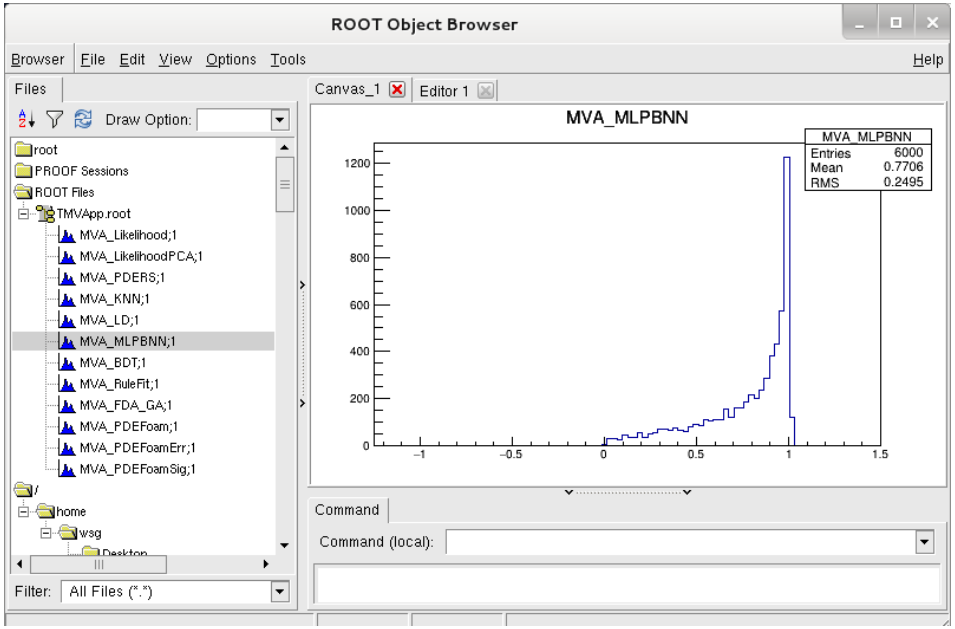
TMVAClassification.C [continue]

```
==> Wrote root file: TMVA.root
==> TMVAClassification is done!
--- Launch TMVA GUI to view input file: TMVA.root
=== Note: inactive buttons indicate classifiers that were not trained, ===
===       or functionalities that were not invoked during the training ===
```



TMVAClassificationApplication.C

```
wsg@debian: ~/work/root/534/tmva/test
File  Edit  View  Search  Terminal  Help
--- Reader                : Booked classifier "RuleFit" of type: "RuleFit"
--- Reader                : Booking "SVM method" of type "SVM" from weights/T
MVAClassification_SVM.weights.xml.
--- MethodBase            : Reading weight file: weights/TMVAClassification_S
VM.weights.xml
--- SVM                   : Read method "SVM" of type "SVM"
--- SVM                   : MVA method was trained with TMVA Version: 4.2.0
--- SVM                   : MVA method was trained with ROOT Version: 5.34/36
--- Reader                : Booked classifier "SVM" of type: "SVM"
--- TMVAClassificationApp : Using input file: http://root.cern.ch/files/tmva_
class_example.root
--- Select signal sample
--- Processing: 6000 events
--- ... Processing event: 0
--- ... Processing event: 1000
--- ... Processing event: 2000
--- ... Processing event: 3000
--- ... Processing event: 4000
--- ... Processing event: 5000
--- End of event loop: Real time 0:06:26, CP time 386.690
--- Created root file: "TMVApp.root" containing the MVA output histograms
==> TMVAClassificationApplication is done!

root [1]
```



```
/**********************************************************************************
 * Project   : TMVA - a Root-integrated toolkit for multivariate data analysis    *
 * Package   : TMVA                                                               *
 * Exectuable: TMVAClassificationApplication                                      *
 *                                                                                *
 * This macro provides a simple example on how to use the trained classifiers     *
 * within an analysis module                                                      *
 **********************************************************************************/

#include <cstdlib>
#include <vector>
#include <iostream>
#include <map>
#include <string>

#include "TFile.h"
#include "TTree.h"
#include "TString.h"
#include "TSystem.h"
#include "TROOT.h"
#include "TStopwatch.h"
```

```
#if not defined(__CINT__) || defined(__MAKECINT__)
#include "TMVA/Tools.h"
#include "TMVA/Reader.h"
#include "TMVA/MethodCuts.h"
#endif

using namespace TMVA;

void TMVAClassificationApplication( TString myMethodList = "" )
{
#ifdef __CINT__
   gROOT->ProcessLine( ".O0" ); // turn off optimization in CINT
#endif

   //---------------------------------------------------------------

   // This loads the library
   TMVA::Tools::Instance();
```

```
// Default MVA methods to be trained + tested
std::map<std::string,int> Use;

// --- Cut optimisation
Use["Cuts"]            = 1;
Use["CutsD"]           = 1;
Use["CutsPCA"]         = 0;
Use["CutsGA"]          = 0;
Use["CutsSA"]          = 0;
//
// --- 1-dimensional likelihood ("naive Bayes estimator")
Use["Likelihood"]      = 1;
Use["LikelihoodD"]     = 0; // the "D" extension indicates decorrelated input variables (see option strings)
Use["LikelihoodPCA"]   = 1; // the "PCA" extension indicates PCA-transformed input variables (see option strings)
Use["LikelihoodKDE"]   = 0;
Use["LikelihoodMIX"]   = 0;
//
// --- Mutidimensional likelihood and Nearest-Neighbour methods
Use["PDERS"]           = 1;
Use["PDERSD"]          = 0;
Use["PDERSPCA"]        = 0;
Use["PDEFoam"]         = 1;
```

```
// --- Create the Reader object

TMVA::Reader *reader = new TMVA::Reader( "!Color:!Silent" );

// Create a set of variables and declare them to the reader
// - the variable names MUST corresponds in name and type to those given in the weight file(s) used
Float_t var1, var2;
Float_t var3, var4;
reader->AddVariable( "myvar1 := var1+var2", &var1 );
reader->AddVariable( "myvar2 := var1-var2", &var2 );
reader->AddVariable( "var3",                &var3 );
reader->AddVariable( "var4",                &var4 );

// Spectator variables declared in the training have to be added to the reader, too
Float_t spec1,spec2;
reader->AddSpectator( "spec1 := var1*2",   &spec1 );
reader->AddSpectator( "spec2 := var1*3",   &spec2 );
```

```
// --- Book the MVA methods

TString dir    = "weights/";
TString prefix = "TMVAClassification";

// Book method(s)
for (std::map<std::string,int>::iterator it = Use.begin(); it != Use.end(); it++) {
   if (it->second) {
      TString methodName = TString(it->first) + TString(" method");
      TString weightfile = dir + prefix + TString("_") + TString(it->first) + TString(".weights.xml");
      reader->BookMVA( methodName, weightfile );
   }
}
```

```
wsg@debian:~/work/root/534/tmva/test/weights$ ll
total 8592
-rw-r--r-- 1 wsg wsg  429018 Jun 14 18:19 TMVAClassification_BDT.class.C
-rw-r--r-- 1 wsg wsg 1764687 Jun 14 18:19 TMVAClassification_BDT.weights.xml
-rw-r--r-- 1 wsg wsg    7886 Jun 14 18:13 TMVAClassification_Cuts.class.C
-rw-r--r-- 1 wsg wsg   12127 Jun 14 18:13 TMVAClassification_CutsD.class.C
-rw-r--r-- 1 wsg wsg   42964 Jun 14 18:13 TMVAClassification_CutsD.weights.xml
-rw-r--r-- 1 wsg wsg   40935 Jun 14 18:13 TMVAClassification_Cuts.weights.xml
-rw-r--r-- 1 wsg wsg    8035 Jun 14 18:16 TMVAClassification_FDA_GA.class.C
-rw-r--r-- 1 wsg wsg    2922 Jun 14 18:16 TMVAClassification_FDA_GA.weights.xml
-rw-r--r-- 1 wsg wsg    7713 Jun 14 18:13 TMVAClassification_KNN.class.C
-rw-r--r-- 1 wsg wsg  925555 Jun 14 18:13 TMVAClassification_KNN.weights.xml
-rw-r--r-- 1 wsg wsg    8106 Jun 14 18:13 TMVAClassification_LD.class.C
-rw-r--r-- 1 wsg wsg    4868 Jun 14 18:13 TMVAClassification_LD.weights.xml
-rw-r--r-- 1 wsg wsg 1350033 Jun 14 18:13 TMVAClassification_Likelihood.class.C
-rw-r--r-- 1 wsg wsg 1342329 Jun 14 18:13 TMVAClassification_LikelihoodPCA.class.C
-rw-r--r-- 1 wsg wsg   15276 Jun 14 18:13 TMVAClassification_LikelihoodPCA.weights.xml
-rw-r--r-- 1 wsg wsg   12993 Jun 14 18:13 TMVAClassification_Likelihood.weights.xml
-rw-r--r-- 1 wsg wsg   17479 Jun 14 18:18 TMVAClassification_MLPBNN.class.C
-rw-r--r-- 1 wsg wsg   81921 Jun 14 18:18 TMVAClassification_MLPBNN.weights.xml
-rw-r--r-- 1 wsg wsg    8190 Jun 14 18:13 TMVAClassification_PDEFoam.class.C
-rw-r--r-- 1 wsg wsg   22121 Jun 14 18:13 TMVAClassification_PDEFoam.weights_foams.root
-rw-r--r-- 1 wsg wsg    3952 Jun 14 18:13 TMVAClassification_PDEFoam.weights.xml
-rw-r--r-- 1 wsg wsg    7788 Jun 14 18:13 TMVAClassification_PDERS.class.C
-rw-r--r-- 1 wsg wsg 1531069 Jun 14 18:13 TMVAClassification_PDERS.weights.xml
-rw-r--r-- 1 wsg wsg   14929 Jun 14 18:19 TMVAClassification_RuleFit.class.C
-rw-r--r-- 1 wsg wsg   26642 Jun 14 18:19 TMVAClassification_RuleFit.weights.xml
-rw-r--r-- 1 wsg wsg  292078 Jun 14 18:19 TMVAClassification_SVM.class.C
-rw-r--r-- 1 wsg wsg  778911 Jun 14 18:19 TMVAClassification_SVM.weights.xml
```

```
// Book output histograms
UInt_t nbin = 100;
TH1F   *histLk(0), *histLkD(0), *histLkPCA(0), *histLkKDE(0), *histLkMIX(0), *histPD(0), *histPDD(0);
TH1F   *histPDPCA(0), *histPDEFoam(0), *histPDEFoamErr(0), *histPDEFoamSig(0), *histKNN(0), *histHm(0);
TH1F   *histFi(0), *histFiG(0), *histFiB(0), *histLD(0), *histNn(0),*histNnbfgs(0),*histNnbnn(0);
TH1F   *histNnC(0), *histNnT(0), *histBdt(0), *histBdtG(0), *histBdtD(0), *histRf(0), *histSVMG(0);
TH1F   *histSVMP(0), *histSVML(0), *histFDAMT(0), *histFDAGA(0), *histCat(0), *histPBdt(0);

if (Use["Likelihood"])    histLk      = new TH1F( "MVA_Likelihood",    "MVA_Likelihood",    nbin, -1, 1 );
if (Use["LikelihoodD"])   histLkD     = new TH1F( "MVA_LikelihoodD",   "MVA_LikelihoodD",   nbin, -1, 0.9999 );
if (Use["LikelihoodPCA"]) histLkPCA   = new TH1F( "MVA_LikelihoodPCA", "MVA_LikelihoodPCA", nbin, -1, 1 );
if (Use["LikelihoodKDE"]) histLkKDE   = new TH1F( "MVA_LikelihoodKDE", "MVA_LikelihoodKDE", nbin,  -0.00001, 0.99999 );
if (Use["LikelihoodMIX"]) histLkMIX   = new TH1F( "MVA_LikelihoodMIX", "MVA_LikelihoodMIX", nbin,  0, 1 );
if (Use["PDERS"])         histPD      = new TH1F( "MVA_PDERS",         "MVA_PDERS",         nbin,  0, 1 );
if (Use["PDERSD"])        histPDD     = new TH1F( "MVA_PDERSD",        "MVA_PDERSD",        nbin,  0, 1 );
if (Use["PDERSPCA"])      histPDPCA   = new TH1F( "MVA_PDERSPCA",      "MVA_PDERSPCA",      nbin,  0, 1 );
```

```
// Prepare input tree (this must be replaced by your data source)
// in this example, there is a toy tree with signal and one with background events
// we'll later on use only the "signal" events for the test in this example.
//
TFile *input(0);
TString fname = "./tmva_example.root";
if (!gSystem->AccessPathName( fname ))
   input = TFile::Open( fname ); // check if file in local directory exists
else
   input = TFile::Open( "http://root.cern.ch/files/tmva_class_example.root" ); // if not: download from ROOT server

if (!input) {
   std::cout << "ERROR: could not open data file" << std::endl;
   exit(1);
}
std::cout << "--- TMVAClassificationApp    : Using input file: " << input->GetName() << std::endl;
```

```
// --- Event loop

// Prepare the event tree
// - here the variable names have to corresponds to your tree
// - you can use the same variables as above which is slightly faster,
//   but of course you can use different ones and copy the values inside the event loop
//
std::cout << "--- Select signal sample" << std::endl;
TTree* theTree = (TTree*)input->Get("TreeS");
Float_t userVar1, userVar2;
theTree->SetBranchAddress( "var1", &userVar1 );
theTree->SetBranchAddress( "var2", &userVar2 );
theTree->SetBranchAddress( "var3", &var3 );
theTree->SetBranchAddress( "var4", &var4 );

// Efficiency calculator for cut method
Int_t    nSelCutsGA = 0;
Double_t effS       = 0.7;

std::vector<Float_t> vecVar(4); // vector for EvaluateMVA tests
```

```
std::cout << "--- Processing: " << theTree->GetEntries() << " events" << std::endl;
TStopwatch sw;
sw.Start();
for (Long64_t ievt=0; ievt<theTree->GetEntries();ievt++) {

   if (ievt%1000 == 0) std::cout << "--- ... Processing event: " << ievt << std::endl;

   theTree->GetEntry(ievt);

   var1 = userVar1 + userVar2;
   var2 = userVar1 - userVar2;

   // --- Return the MVA outputs and fill into histograms

   if (Use["CutsGA"]) {
      // Cuts is a special case: give the desired signal efficienciy
      Bool_t passed = reader->EvaluateMVA( "CutsGA method", effS );
      if (passed) nSelCutsGA++;
   }

   if (Use["Likelihood"    ])   histLk     ->Fill( reader->EvaluateMVA( "Likelihood method"    ) );
```

```
eader = new TMVA::Reader( "!Color:!Silent" );

ate a set of variables and declare them to the reader
- the variable names MUST corresponds in name and type to those given in the weight file(s)
loat_t var1, var2;
Float_t var3, var4;
reader->AddVariable( "myvar1 := var1+var2", &var1 );
reader->AddVariable( "myvar2 := var1-var2", &var2 );
reader->AddVariable( "var3",                &var3 );
ader->AddVariable( "var4",                &var4 );

tor variables declared in the training have to be added to the reader, too
pec2;
"spec1 := var1*2",   &spec1 );
                     &spec2 );
```

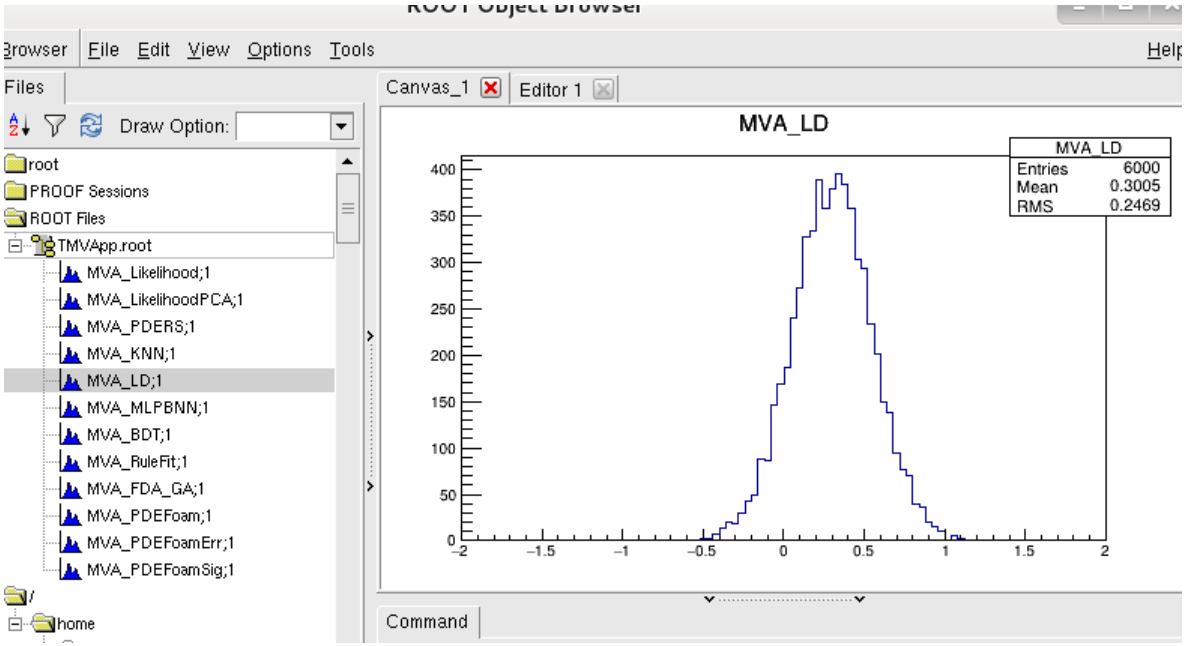
Reader 内部变量有var1。。4值作为计算的输入

TMVAClassification_Likelihood.weights.xml :

```
 <Variables NVar="4">
   <Variable VarIndex="0" Expression="var1+var2" Label="myvar1" Title="myvar1" Unit="" Internal="myvar1" Type="F" Min="-8.96602058
-00" Max="7.69307804e+00"/>
   <Variable VarIndex="1" Expression="var1-var2" Label="myvar2" Title="Expression 2" Unit="" Internal="myvar2" Type="F" Min="-4.08
10010e+00" Max="4.02589369e+00"/>
   <Variable VarIndex="2" Expression="var3" Label="var3" Title="Variable 3" Unit="units" Internal="var3" Type="F" Min="-5.05078554
-00" Max="4.35967064e+00"/>
   <Variable VarIndex="3" Expression="var4" Label="var4" Title="Variable 4" Unit="units" Internal="var4" Type="F" Min="-5.95050764
-00" Max="4.92254400e+00"/>
 </Variables>
```

```
if (Use["Likelihood"    ])   histLk     ->Fill( reader->EvaluateMVA( "Likelihood method"     ) );
if (Use["LikelihoodD"   ])   histLkD    ->Fill( reader->EvaluateMVA( "LikelihoodD method"    ) );
if (Use["LikelihoodPCA"])    histLkPCA  ->Fill( reader->EvaluateMVA( "LikelihoodPCA method"  ) );
if (Use["LikelihoodKDE"])    histLkKDE  ->Fill( reader->EvaluateMVA( "LikelihoodKDE method"  ) );
if (Use["LikelihoodMIX"])    histLkMIX  ->Fill( reader->EvaluateMVA( "LikelihoodMIX method"  ) );
if (Use["PDERS"         ])   histPD     ->Fill( reader->EvaluateMVA( "PDERS method"          ) );
if (Use["PDERSD"        ])   histPDD    ->Fill( reader->EvaluateMVA( "PDERSD method"         ) );
if (Use["PDERSPCA"      ])   histPDPCA  ->Fill( reader->EvaluateMVA( "PDERSPCA method"       ) );
if (Use["KNN"           ])   histKNN    ->Fill( reader->EvaluateMVA( "KNN method"            ) );
if (Use["HMatrix"       ])   histHm     ->Fill( reader->EvaluateMVA( "HMatrix method"        ) );
if (Use["Fisher"        ])   histFi     ->Fill( reader->EvaluateMVA( "Fisher method"         ) );
if (Use["FisherG"       ])   histFiG    ->Fill( reader->EvaluateMVA( "FisherG method"        ) );
if (Use["BoostedFisher"])    histFiB    ->Fill( reader->EvaluateMVA( "BoostedFisher method"  ) );
if (Use["LD"            ])   histLD     ->Fill( reader->EvaluateMVA( "LD method"             ) );
```

```
$ root TMVApp.root
root [0]
Attaching file TMVApp.root as
_file0...
root [1] new TBr

TBrowser
TBrowserImp
TBrowserTimer
TBrowser::
TBranch
root [1] new TBrowser
(class TBrowser*)0xca9df0
```



siguang@pku.edu.cn

TMVA::Reader

```
//#class TMVA::Reader: public TMVA::Configurable
/*
  The Reader class serves to use the MVAs in a specific analysis context.
  Within an event loop, a vector is filled that corresponds to the variables
  that were used to train the MVA(s) during the training stage. This vector
  is transfered to the Reader, who takes care of interpreting the weight
  file of the MVA of choice, and to return the MVA's output. This is then
  used by the user for further analysis.


  Usage:
*/
```

```
void read(){
  // ------ before starting the event loop (eg, in the initialisation step)


    // create TMVA::Reader object

    TMVA::Reader *reader = new TMVA::Reader();

    // create a set of variables and declare them to the reader
    // - the variable names must corresponds in name and type to
    // those given in the weight file(s) that you use
    Float_t var1, var2, var3, var4;
    reader->AddVariable( "var1", &var1 );
    reader->AddVariable( "var2", &var2 );
    reader->AddVariable( "var3", &var3 );
    reader->AddVariable( "var4", &var4 );

    // book the MVA of your choice (prior training of these methods, ie,
    // existence of the weight files is required)
    reader->BookMVA( "Fisher method",  "weights/Fisher.weights.txt"   );
    reader->BookMVA( "MLP method",     "weights/MLP.weights.txt" );
```

```
   // ------- start your event loop

   for (Long64_t ievt=0; ievt<myTree->GetEntries();ievt++) {

     // fill vector with values of variables computed from those in the tree
     var1 = myvar1;
     var2 = myvar2;
     var3 = myvar3;
     var4 = myvar4;

     // retrieve the corresponding MVA output
     double mvaFi = reader->EvaluateMVA( "Fisher method" );
     double mvaNN = reader->EvaluateMVA( "MLP method"    );

     // do something with these ...., e.g., fill them into your ntuple

   } // end of event loop

    delete reader;
}
```