章：图表

- 6个图
  - 6.1 TGraph
  - 6.2叠加两个图
  - 6.3带误差线的图形
  - 6.4具有不对称误差条的图
  - 6.5具有不对称弯曲误差的图
  - 6.6 TGraphPolar
  - 6.7 TGraph禁区
  - 6.8 TGraphQQ
  - 6.9 TMultiGraph
  - 6.10 TGraph2D
  - 6.11 TGraph2DErrors
  - 6.12拟合图表
  - 6.13设置图形的轴标题
  - 6.14缩放图表
  - 6.15图形的用户界面

6个图

图形是由两个数组X和Y组成的图形对象，保持点的x，y坐标 `n` 。有几个图表类; 它们是 `TGraph` ， `TGraphErrors` ， `TGraphAsymmErrors` ，和 `TMultiGraph` 。

6.1 TGraph

本 `TGraph` 类支持非等距点，一般情况下，并与等距点的特殊情况。使用 `TGraph` 构造函数创建图形。首先，我们定义坐标数组，然后创建图形。坐标可以是双精度数或浮点数。

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

另一种构造函数只占用点数 `n` 。预计坐标将在稍后设置。

```
TGraph *gr2 = new TGraph(n);
```

也可以使用默认构造函数。进一步调用 `SetPoint()` 将扩展内部向量。

```
TGraph *gr3 = new TGraph();
```

6.1.1图形绘制选项

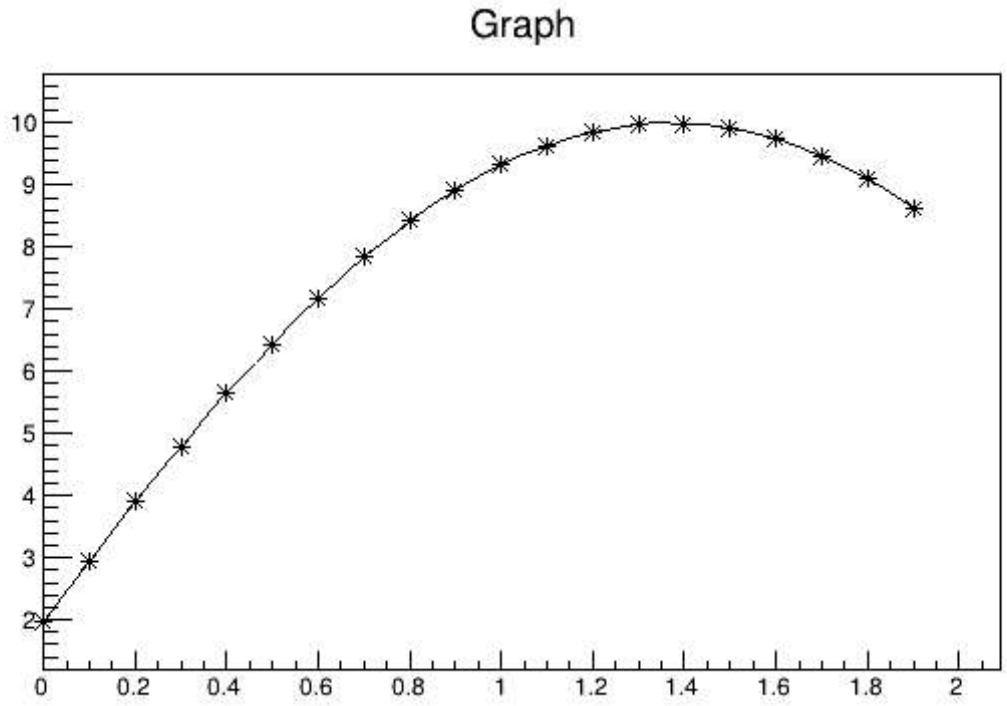
图中的各种绘图选项将在下面进行说明 `TGraph::PaintGraph` 。他们是：

- “ `L` ”绘制每个点之间的简单多边形线
- “ `F` ”绘制填充区域
- “ `F1` ”同上为“ `F` ”但填充区域不再是X = 0或Y = 0的响应区域
- “ `F2` ”绘制连接箱子中心的填充区域多边形线
- “ `A` ”围绕图形绘制轴
- “ `C` ”画出一条平滑的曲线

- “ `*` ”每个点都绘制一个星
- “ `P` ”图表的当前标记绘制在每个点
- “ `B` ”每个点都绘制一个条形图
- “ `[]` ”仅绘制误差条的末端垂直/水平线。此选项仅适用于 `TGraphAsymmErrors` 。
- “ `1` ” `ylow = rwymin`

这些选项不区分大小写，在大多数情况下它们可以连接在一起。我们来看一些例子。

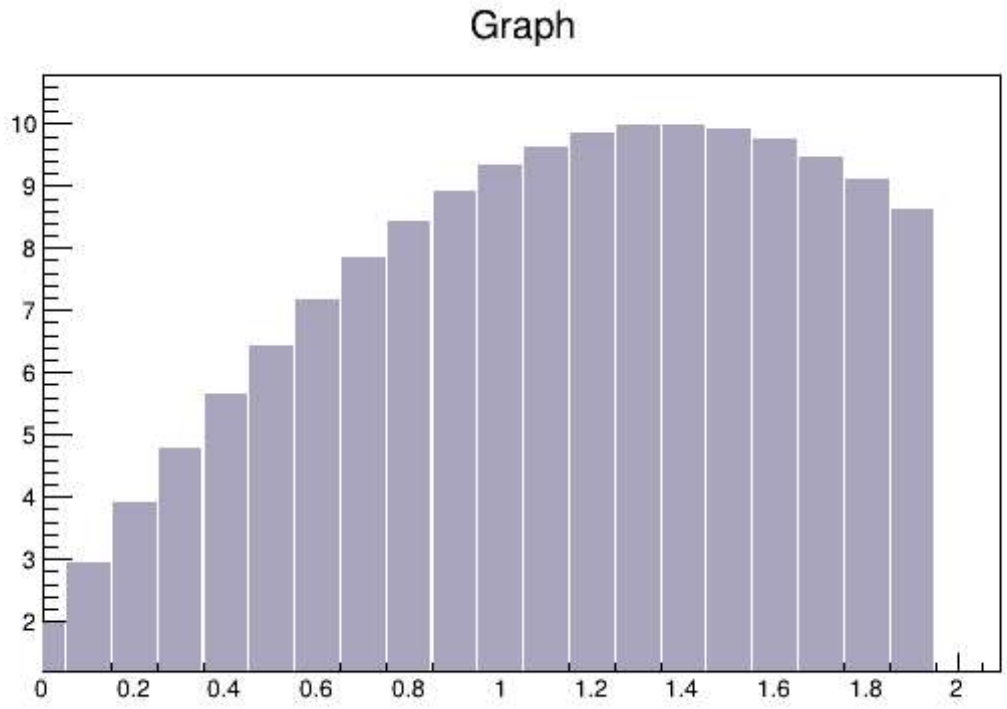
6.1.1.1连续线，轴和星（AC *）



用轴，*标记和实线绘制的图形（选项AC *）

```
{
   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);
   }

   // create graph
   TGraph *gr  = new TGraph(n,x,y);
   TCanvas *c1 = new TCanvas("c1","Graph Draw Options",
                             200,10,600,400);

   // draw the graph with axis, continuous line, and put
   // a * at each point
   gr->Draw("AC*");
}
```

6.1.1.2条形图（AB）

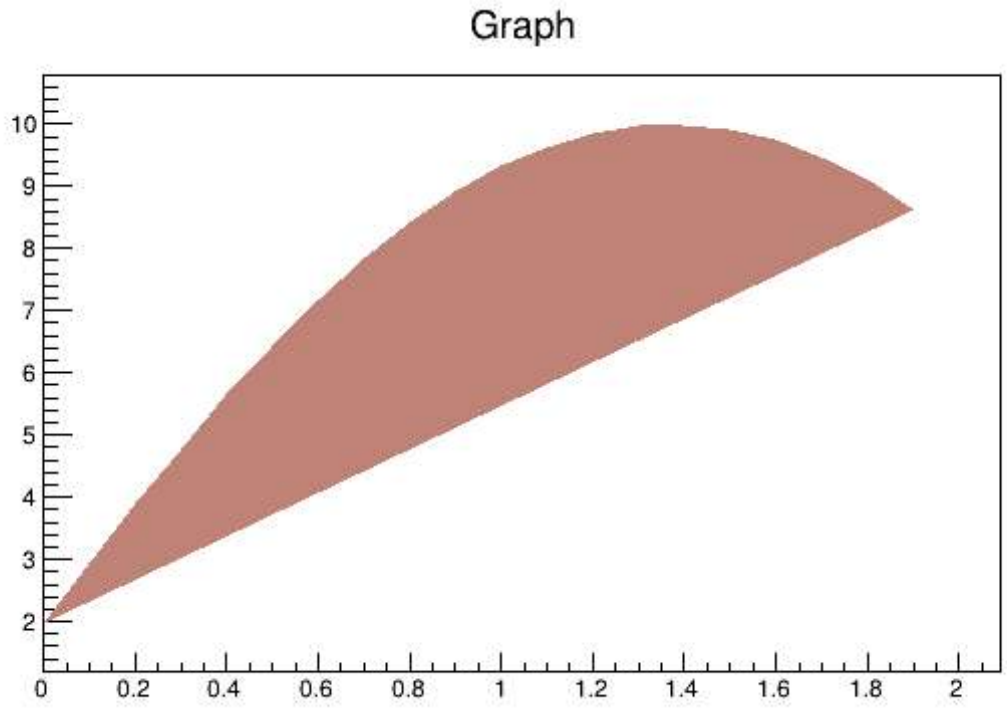


用轴和条绘制的图形（选项AB）

```
root[] TGraph *gr1 = new TGraph(n,x,y);
root[] gr1->SetFillColor(40);
root[] gr1->Draw("AB");
```

只有在定义了n，x和y时，此代码才有效。前面的示例定义了这些。您需要设置填充颜色，因为默认情况下填充颜色为白色，并且在白色画布上不可见。您还需要为其指定一个轴，否则条形图将无法正确显示。

6.1.1.3填充图（AF）

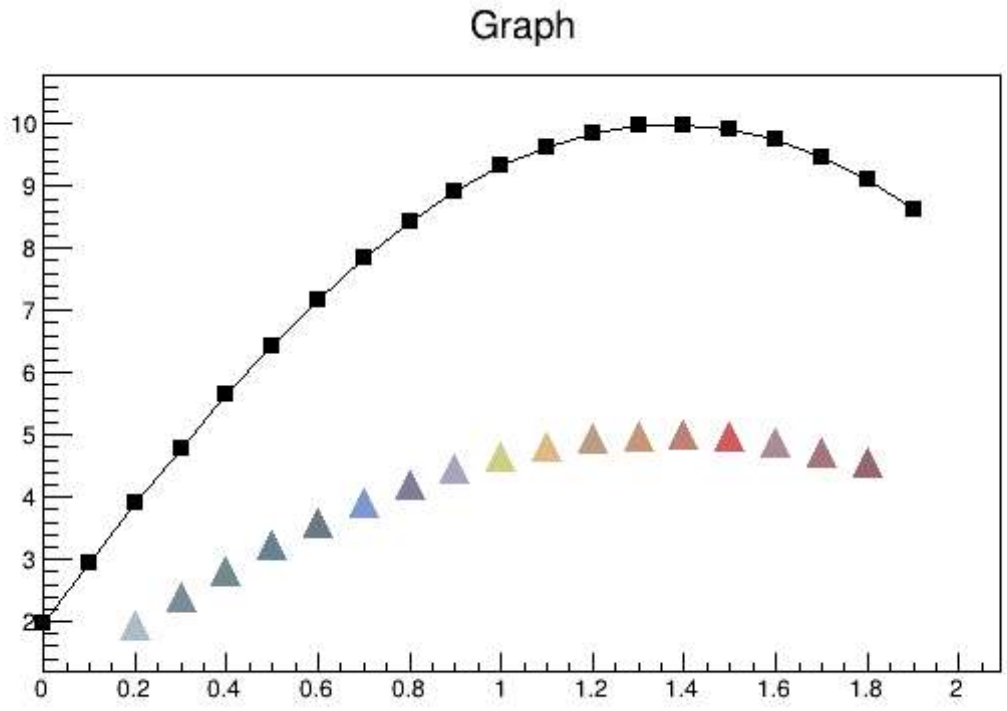


使用轴和填充绘制的图形（选项AF）

```
root[] TGraph *gr3 = new TGraph(n,x,y);
root[] gr3->SetFillColor(45);
root[] gr3->Draw("AF")
```

如果此代码只会工作 `n` ， `x` ， `y` 定义。第一个例子定义了它们。您需要设置填充颜色，因为默认情况下填充颜色为白色，并且在白色画布上不可见。您还需要为其指定一个轴，否则填充的多边形将无法正确显示。

6.1.1.4标记选项

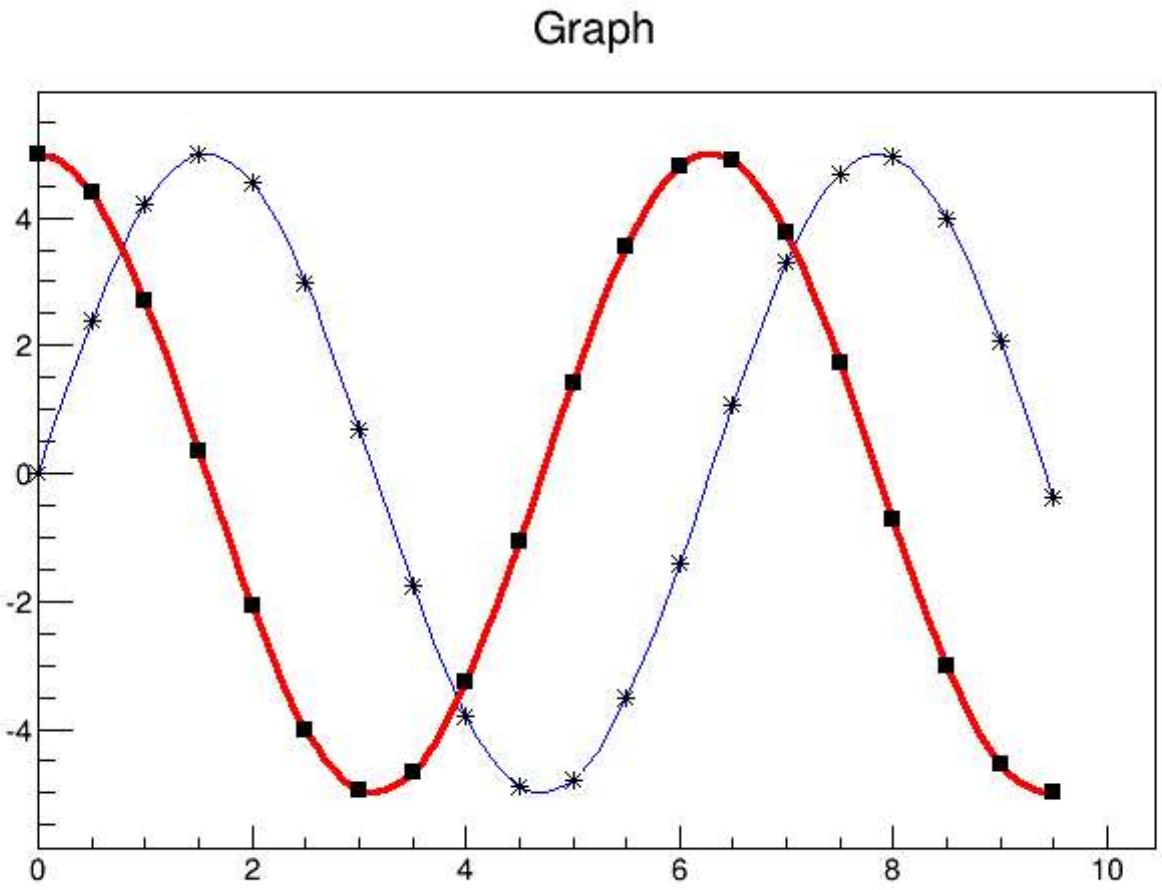


以不同方式创建的图形标记

```
{
   Int_t n = 20;
   Double_t x[n], y[n];

   // build the arrays with the coordinate of points
   for (Int_t i=0; i<n; i++) {
      x[i] = i*0.1;
      y[i] = 10*sin(x[i]+0.2);
   }

   // create graphs
   TGraph *gr3  = new TGraph(n,x,y);
   TCanvas *c1 = new TCanvas ("c1","Graph Draw Options",
                              200,10,600,400);

   // draw the graph with the axis,contineous line, and put
   // a marker using the graph's marker style at each point
   gr3->SetMarkerStyle(21);
   c1->cd(4);
   gr3->Draw("APL");

   // get the points in the graph and put them into an array
   Double_t *nx = gr3->GetX();
   Double_t *ny = gr3->GetY();

   // create markers of different colors
   for (Int_t j=2; j<n-1; j++) {
      TMarker *m = new TMarker(nx[j], 0.5*ny[j], 22);
      m->SetMarkerSize(2);
      m->SetMarkerColor(31+j);
      m->Draw();
   }
}
```

6.2叠加两个图

要超级施加两个图形，您只需要绘制一次轴，并在第二个图形的绘制选项中省略“A”。接下来是一个例子：



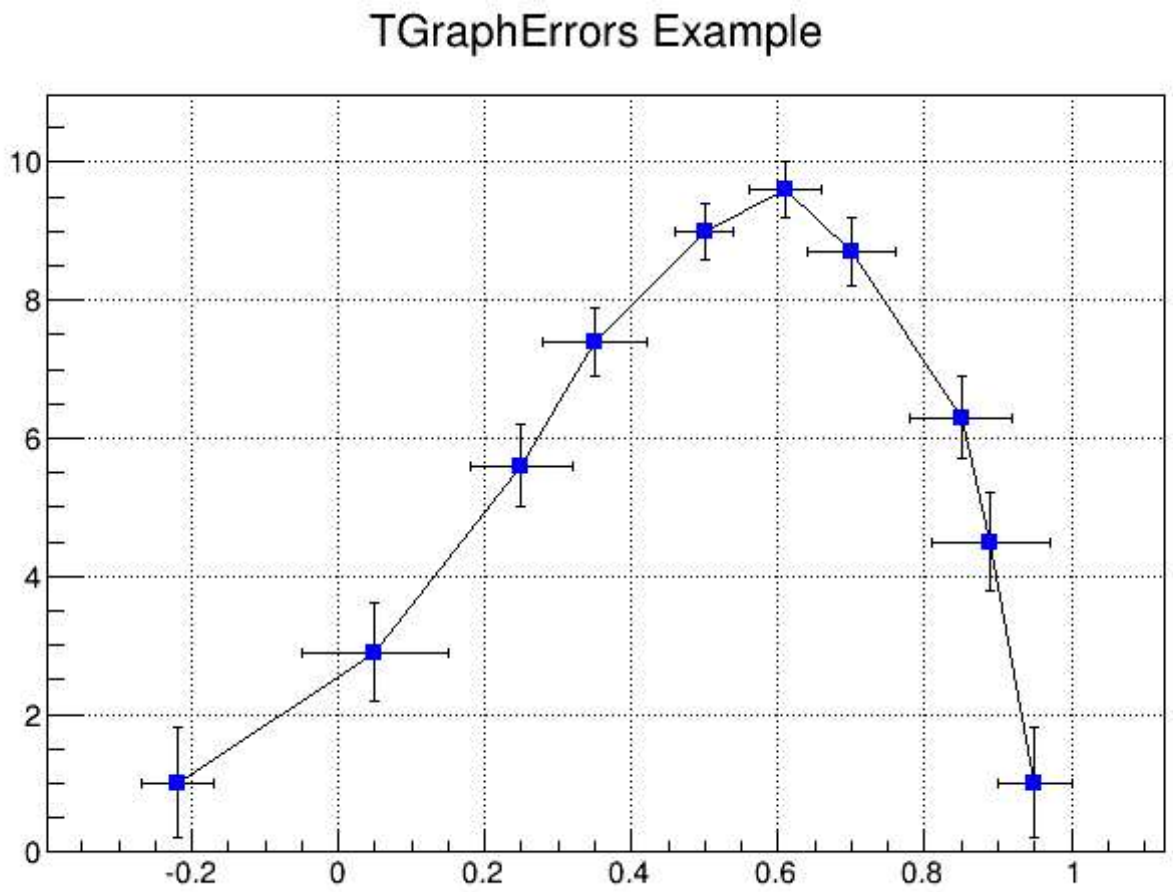
叠加两个图

```
{
   Int_t n = 20;
   Double_t x[n], y[n], x1[n], y1[n];

   // create a blue graph with a cos function
   gr1->SetLineColor(4);
   gr1->Draw("AC*");

   // superimpose the second graph by leaving out the axis option "A"
   gr2->SetLineWidth(3);
   gr2->SetMarkerStyle(21);
   gr2->SetLineColor(2);
   gr2->Draw("CP");
}
```

6.3带误差线的图形

A `TGraphErrors` 是 `TGraph` 带有误差线的。各种绘制格式选项 `TGraphErrors::Paint()` 源自 `TGraph` 。

```
void TGraphErrors::Paint(Option_t *option)
```



具有错误条的不同绘制选项的图形

此外，可以使用“ `Z` ”选项绘制它，以便在误差线的末尾留下小线条。如果选项包含“ `>` ”，则会在错误栏的末尾绘制一个箭头。如果选项包含“ `|>` ”，则会在错误栏的末尾绘制完整箭头。箭头的大小设置为标记大小的2/3。

选项“ `[]` ”有趣的是将系统误差叠加在图表顶部并带有统计误差。指定时，仅绘制误差线的末端垂直/水平线。

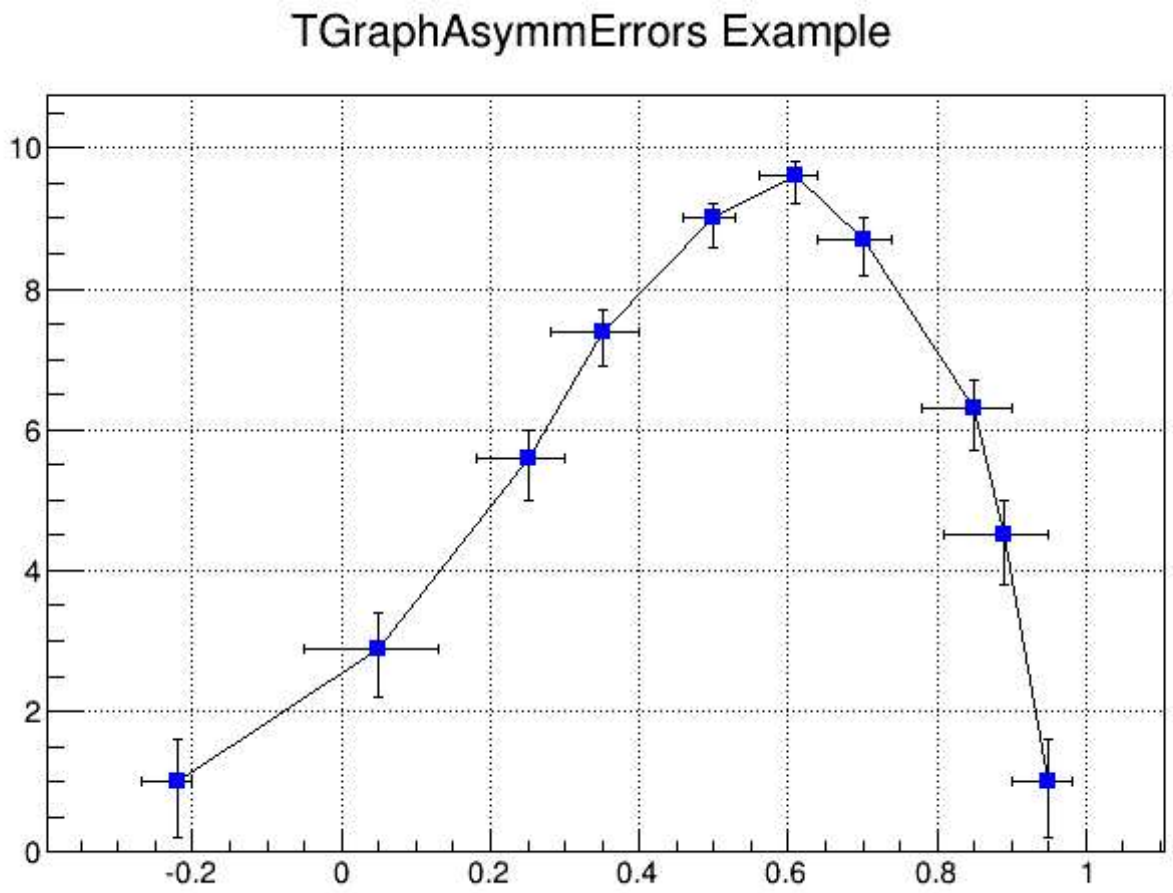
要控制误差线末尾的线条大小（选择选项1时），请使用 `SetEndErrorSize(np)` 。默认情况下 `np=1` ； `np` 表示像素数。

```
gStyle->SetEndErrorSize(np);
```

这四个参数 `TGraphErrors` 是:(`X, Y` 如 `TGraph`) , - 错误 `X` 和 `Y` - 错误 - `x` 和 `y` 方向错误的大小。下一个例子是 `$ROOTSYS/tutorials/graphs/gerrors.C.`

```
{
   c1 = new TCanvas("c1","A Simple Graph with error bars",200,10,700,500);
   c1->SetGrid();

   // create the coordinate arrays
   Int_t n = 10;
   Float_t x[n]  = {-.22,.05,.25,.35,.5,.61,.7,.85,.89,.95};
   Float_t y[n]  = {1,2.9,5.6,7.4,9,9.6,8.7,6.3,4.5,1};

   // create the error arrays
   Float_t ex[n] = {.05,.1,.07,.07,.04,.05,.06,.07,.08,.05};
   Float_t ey[n] = {.8,.7,.6,.5,.4,.5,.6,.7,.8};

   // create the TGraphErrors and draw it
   gr = new TGraphErrors(n,x,y,ex,ey);
   gr->SetTitle("TGraphErrors Example");
   gr->SetMarkerColor(4);
   gr->SetMarkerStyle(21);
   gr->Draw("ALP");
   c1->Update();
}
```

6.4具有不对称误差条的图



具有不对称误差条的图形

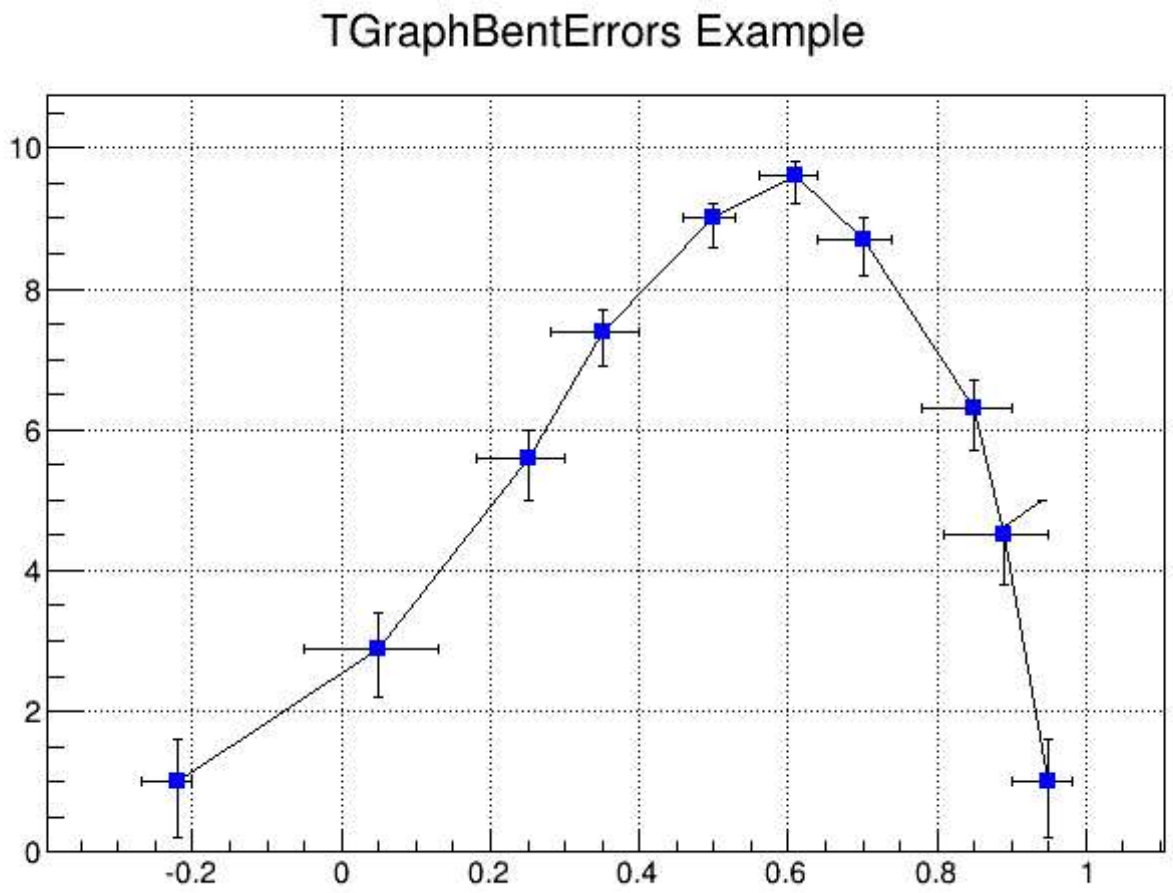
A `TGraphAsymmErrors` 是 `TGraph` 具有不对称误差条的a。它继承了各种绘制格式选项 `TGraph` 。它的方法 `Paint(Option_t *option)` 用 `TGraphAsymmErrors` 当前属性绘制。您可以为绘图设置以下附加选项：

- “ `z` ”或“ `Z` ”水平和垂直小线不会在误差线的末尾绘制
- “ `>` ”在误差线的末尾绘制一个箭头
- “ `|>` ”在错误栏的末尾绘制一个完整的箭头; 它的大小是 $\frac{2}{3}$ – 标记大小
- “ `[]` ”仅绘制误差条的末端垂直/水平线; 这个选项很有意思，可以在具有统计误差的图表上叠加系统误差。

构造函数有六个数组作为参数：X和Y为**TGraph**，低X错误和高X错误，低Y错误和高Y错误。低值是向左和向下的误差条的长度，高值是向右和向上的误差条的长度。

```
{
   c1 = new TCanvas("c1","A Simple Graph with error bars",
                    200,10,700,500);
   c1->SetGrid();

   // create the arrays for the points
   Int_t n = 10;
   Double_t x[n]  = {-.22,.05,.25,.35,.5, .61,.7,.85,.89,.95};
   Double_t y[n]  = {1,2.9,5.6,7.4,9,9.6,8.7,6.3,4.5,1};

   // create the arrays with high and low errors
   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};

   // create TGraphAsymmErrors with the arrays
   gr = new TGraphAsymmErrors(n,x,y,exl,exh,eyl,eyh);
   gr->SetTitle("TGraphAsymmErrors Example");
   gr->SetMarkerColor(4);
   gr->SetMarkerStyle(21);
   gr->Draw("ALP");
}
```

6.5具有不对称弯曲误差的图



具有不对称弯曲误差条的图形

A `TGraphBentErrors` 是 `TGraph` 具有弯曲的，不对称的误差条。绘制a的各种格式选项 `TGraphBentErrors` 在 `TGraphBentErrors::Paint` **方法** `TGraphBentErrors` 中解释。**在**默认情况下，用误差棒和小水平线和垂直线的误差棒的端部引出。如果指定选项“ `z` ”或“ `Z` ”，则不绘制这些小线。如果 `X` 指定选项“ ”，则不绘制错误（ `TGraph::Paint` 等效方法）。

- 如果选项包含“ `>` ”，则会在错误栏的末尾绘制一个箭头
- 如果选项包含“ `|>` ”，则会在错误栏的末尾绘制一个完整的箭头
- 箭头的大小设置为标记大小的2/3

- 如果 `[]` 指定选项“ ”，则仅绘制误差线的末端垂直/水平线。这个选项很有意思，可以将系统误差叠加在具有统计误差的图表之上。

此图由以下宏生成：

```
{
   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,.0,.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};
   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");
}
```

6.6 TGraphPolar

在 `TGraphPolar` 类创建极线图（包括误差条）。A `TGraphPolar` 是以 `TGraphErrors` 极坐标表示的。它使用类 `TGraphPolargram` 绘制极轴。

```
{
   TCanvas *CPol = new TCanvas("CPol","TGraphPolar Examples",700,700);
   Double_t rmin=0;
   Double_t rmax=TMath::Pi()*2;
   Double_t r[1000];
   Double_t theta[1000];
   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->SetLineColor(2);
   grP1->Draw("AOL");
}
```

TGraphPolar绘图选项包括：

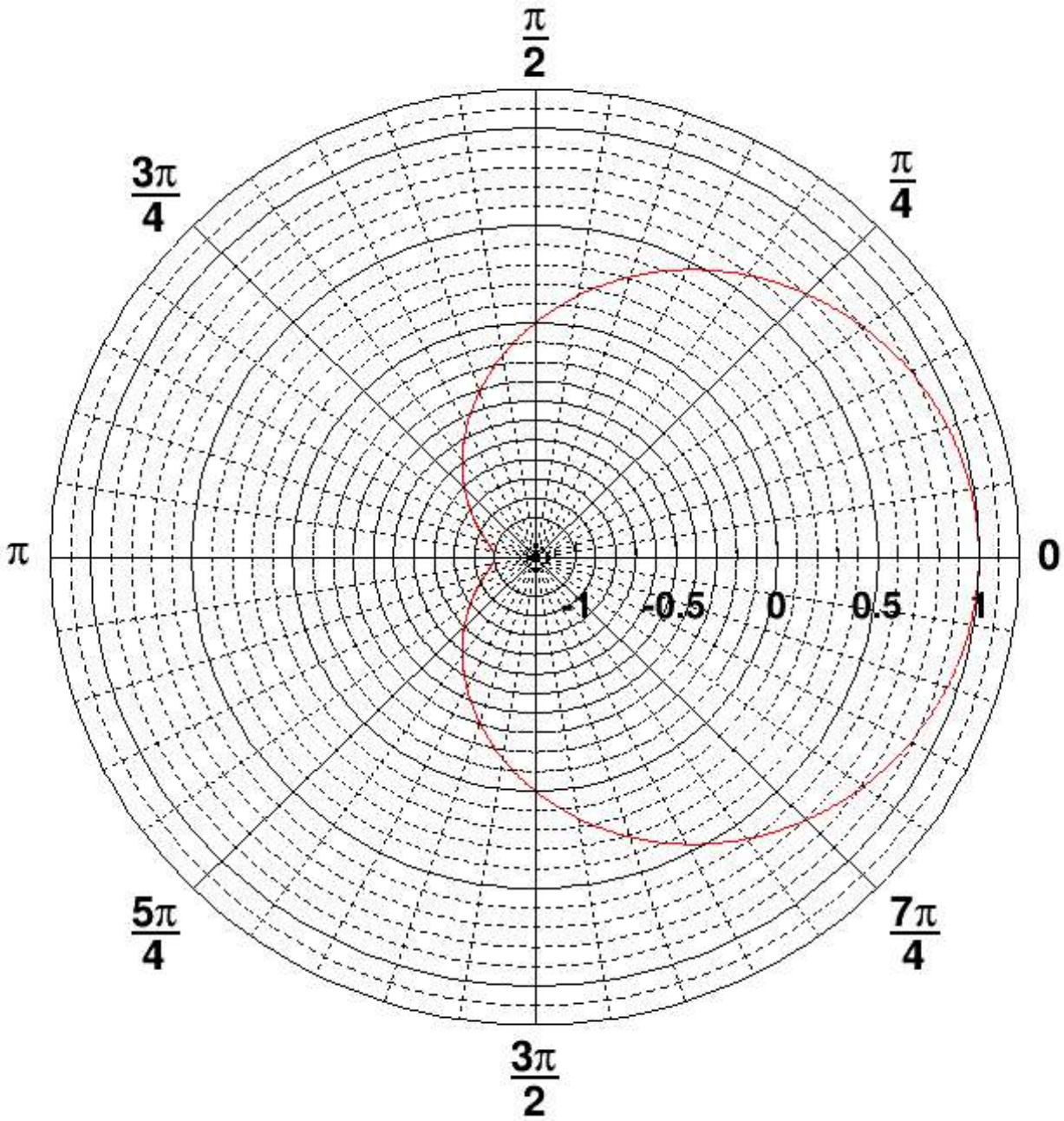
“O”极性标签与极谱半径正交。

“P”Polymarker在每个点位置涂漆。

“E”画错误条。

“F”绘制填充区域（闭合多边形）。

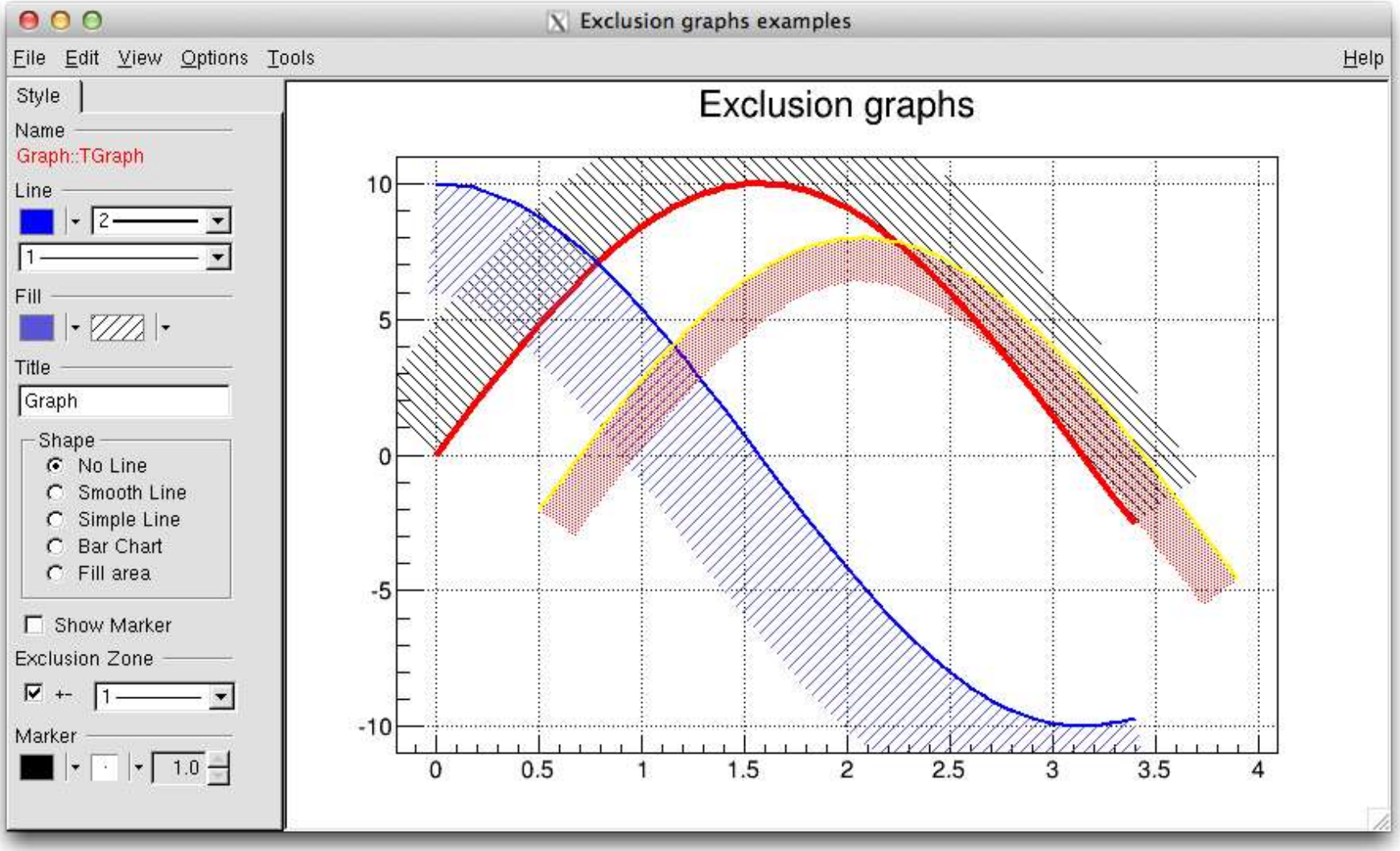
即使已存在极化，“A”强制轴重绘。



极坐标图

6.7 TGraph禁区

使用选项“ C ”或“ L ”绘制图形时，可以在线条的一侧绘制填充区域。这对于显示禁区很有用。当图线宽度的绝对值（由于设置 SetLineWidth ）大于99时，此绘图模式被激活。在这种情况下，线宽编号被解释为 100*ff+ll = ffll 。两位数字“ ll ”表示法线宽度，而“ ff ”表示填充区域宽度。“ ffll ”符号允许将填充区域从线的一侧翻转到另一侧。当前填充区域属性用于绘制阴影区域。



带有禁区的图表

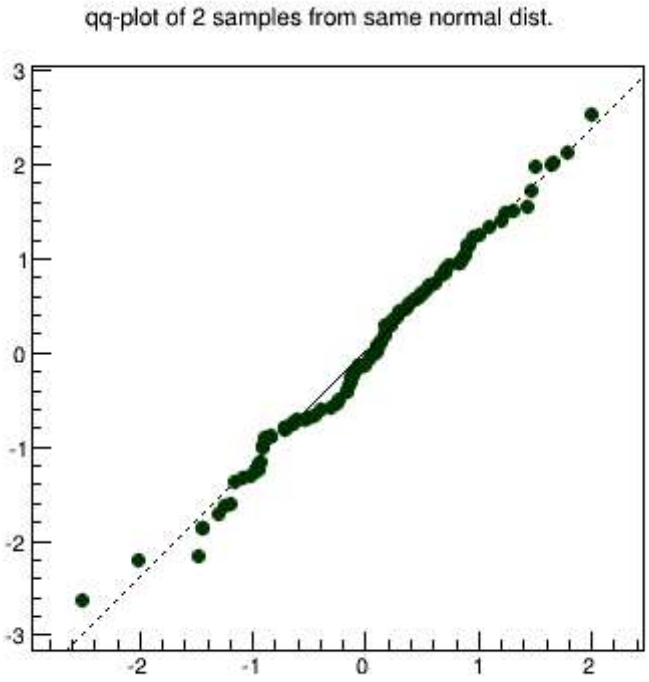
```
{
   c1 = new TCanvas("c1","Exclusion graphs examples",200,10,700,500);
   c1->SetGrid();

   // create the multigraph
   TMultiGraph *mg = new TMultiGraph();
   mg->SetTitle("Exclusion graphs");

   // create the graphs points
   const Int_t n = 35;
   Double_t x1[n], x2[n], x3[n], y1[n], y2[n], y3[n];
   for (Int_t i=0;i<n;i++) {
      x1[i] = i*0.1; y1[i] = 10*sin(x1[i]);
      x2[i] = x1[i]; y2[i] = 10*cos(x1[i]);
      x3[i] = x1[i]+.5; y3[i] = 10*sin(x1[i])-2;
   }

   // create the 1st TGraph
   gr1 = new TGraph(n,x1,y1);
   gr1->SetLineColor(2);
   gr1->SetLineWidth(1504);
   gr1->SetFillStyle(3005);

   // create the 2nd TGraph
   gr2 = new TGraph(n,x2,y2);
   gr2->SetLineColor(4);
   gr2->SetLineWidth(-2002);
   gr2->SetFillStyle(3004);
   gr2->SetFillColor(9);

   // create the 3rd TGraph
   gr3 = new TGraph(n,x3,y3);
   gr3->SetLineColor(5);
   gr3->SetLineWidth(-802);
   gr3->SetFillStyle(3002);
   gr3->SetFillColor(2);
```

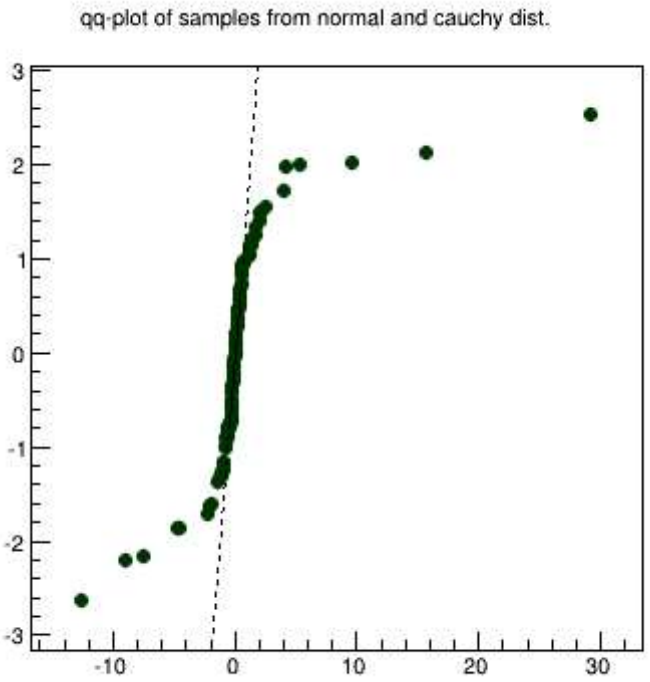
```
   // put the graphs in the multigraph
   mg->Add(gr1);
   mg->Add(gr2);
   mg->Add(gr3);

   // draw the multigraph
   mg->Draw("AC");
}
```

6.8 TGraphQQ

A `TGraphQQ` 允许绘制分位数 - 分位数图。可以为两个数据集绘制这样的图，或者针对一个数据集和理论分布函数绘制这样的图。

6.8.1两个数据集



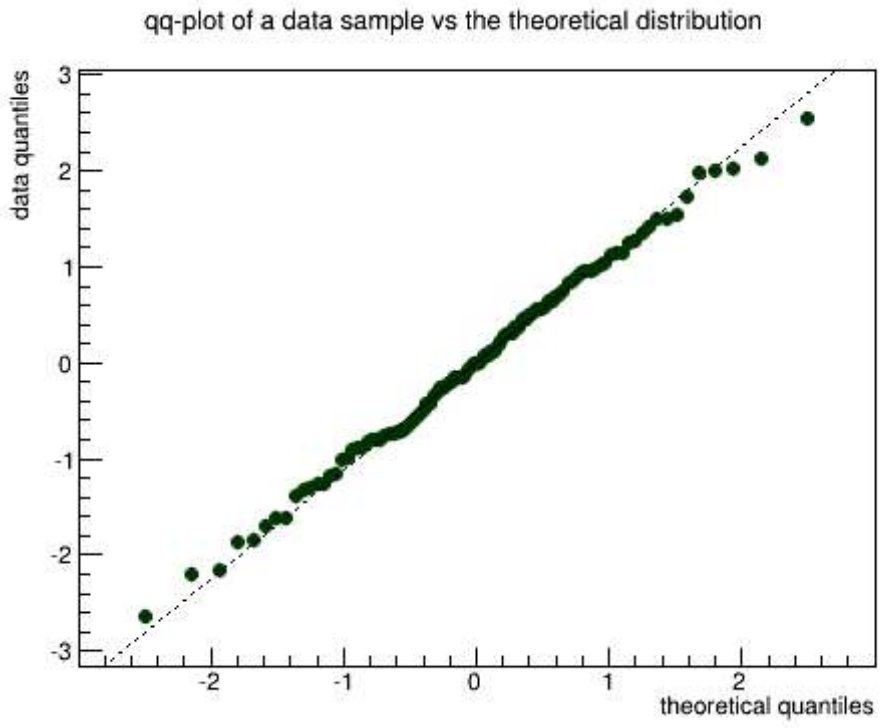


2个数据集的qq图的示例

分位数 - 分位数图用于确定两个样本是否来自相同分布。qq-plot绘制一个数据集的分位数与另一个数据集的分位数。具有较少条目的数据集的分位数在Y轴上，具有更多条目 - 在X轴上。还绘制了通过0.25和0.75分位数的直线以供参考。它表示强大的线性拟合，对数据集的极端不敏感。如果数据集来自同一分布，则绘图的点应大致落在45度线上。如果它们具有相同的分布函数，但位置或比例的参数不同，它们仍然应该落在直线上，而不是45度。

他们越偏离直线，越有证据表明数据集来自不同的分布。qq-plot的优势在于它不仅表明底层分布不同，而且与分析方法不同，它还提供了有关这种差异性质的信息：较重的尾部，不同的位置/比例，不同的形状等。

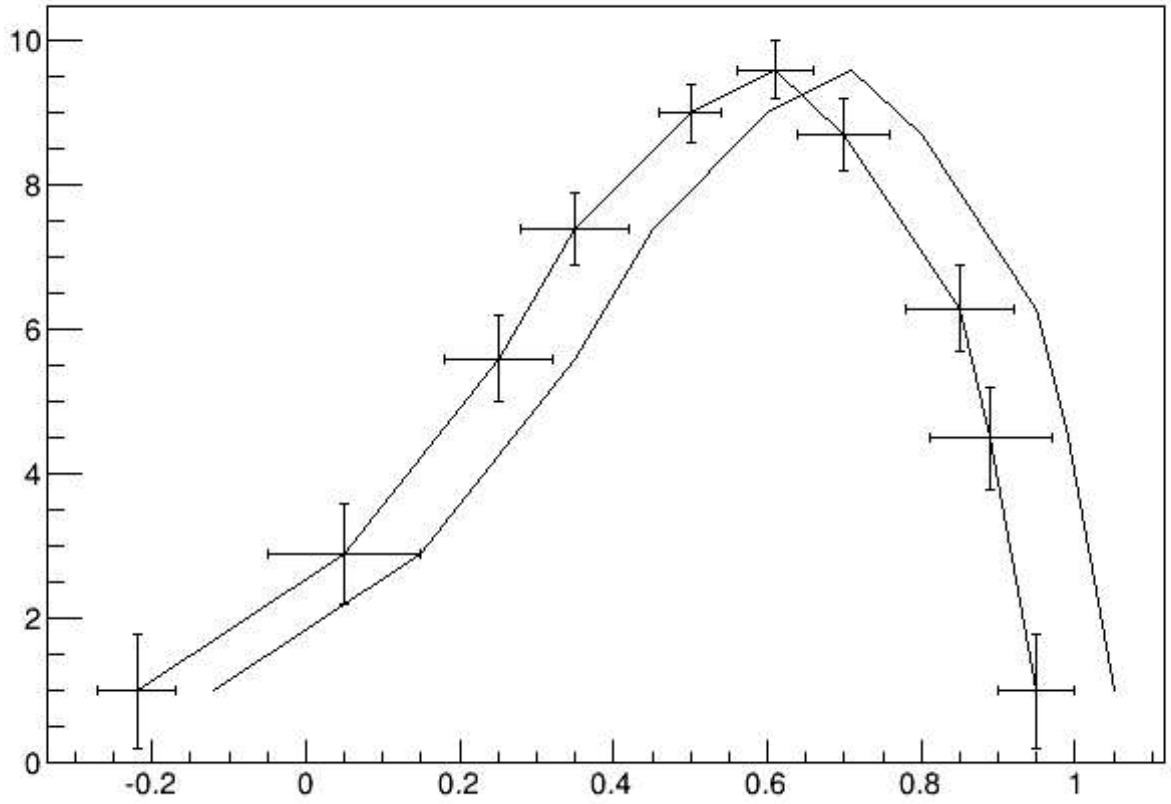
6.8.2一个数据集



1个数据集的qq图的示例

分位数分位数图用于确定数据集是否来自指定的理论分布，例如正态分布。qq-plot绘制数据集的分位数与指定理论分布的分位数。注意，密度，而不是CDF应该指定为直线，也可以绘制0.25和0.75分位数作为参考。它表示稳健的线性拟合，对数据集的极值不敏感。与两个数据集情况一样，偏离直线表示偏离指定的分布。下图显示了来自N（3,2）分布和TMath :: Gaus（0,1）理论函数的数据集的qq图的示例。拟合参数是分布均值和西格玛的估计。

6.9 TMultiGraph

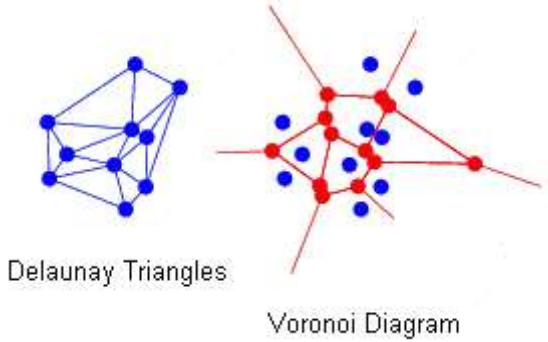


一个多图的例子

A `TMultiGraph` 是 `TGraph` （或派生的）对象的集合。用于 `TMultiGraph::Add` 向列表中添加新图表。该 `TMultiGraph` 公司拥有该列表中的对象。绘图和拟合选项与for相同 `TGraph` 。

```
{
   // create the points
   Int_t n = 10;
   Double_t x[n]  = {-.22,.05,.25,.35,.5,.61,.7,.85,.89,.95};
   Double_t y[n]  = {1,2.9,5.6,7.4,9,9.6,8.7,6.3,4.5,1};
   Double_t x2[n]  = {-.12,.15,.35,.45,.6,.71,.8,.95,.99,1.05};
   Double_t y2[n]  = {1,2.9,5.6,7.4,9,9.6,8.7,6.3,4.5,1};

   // create the width of errors in x and y direction
   Double_t ex[n] = {.05,.1,.07,.07,.04,.05,.06,.07,.08,.05};
   Double_t ey[n] = {.8,.7,.6,.5,.4,.4,.5,.6,.7,.8};

   // create two graphs
   TGraph *gr1 = new TGraph(n,x2,y2);
   TGraphErrors *gr2 = new TGraphErrors(n,x,y,ex,ey);

   // create a multigraph and draw it
   TMultiGraph  *mg  = new TMultiGraph();
   mg->Add(gr1);
   mg->Add(gr2);
   mg->Draw("ALP");
}
```

6.10 TGraph2D



Delaunay三角形和Voronoï图

这个类是一组 N 点 x[i] ， y[i] ， z[i] 在非均匀网格。实现了几种可视化技术，包括Delaunay三角剖分。Delaunay三角剖分定义如下：'对于 S 欧几里德平面中的一组点，这是一个独特的三角剖分 DT(S) ， S 使得 S 任何三角形的圆周内没有任何点 DT(S) 。 DT(S) 是Voronoï图的双重图 S 。如果n是点数 S ，则S的Voronoï图是包含平面的分区 S 指向n个凸多边形，使得每个多边形恰好包含一个点，并且给定多边形中的每个点都更接近其中心点而不是任何其他点。Voronoï图有时也称为Dirichlet曲面细分。

本 TGraph2D 类具有以下构造函数：

- 使用数组的维度 n 和三个数组 x ， y 和 z （可以是双精度数，浮点数或整数数组）：

```
TGraph2D *g = new TGraph2D(n,x,y,z);
```

- 仅限数组维度：

```
TGraph2D *g = new TGraph2D(n);
```

- 内部阵列填充有方法 SetPoint 在位置“ i ”与值 x ， y ， z ：

```
g->SetPoint(i,x,y,z);
```

- 没有参数； SetPoint 必须使用该方法来填充内部数组。

```
TGraph2D *g = new TGraph2D();
```

- 从文件：

```
TGraph2D *g = new TGraph2D("graph.dat");
```

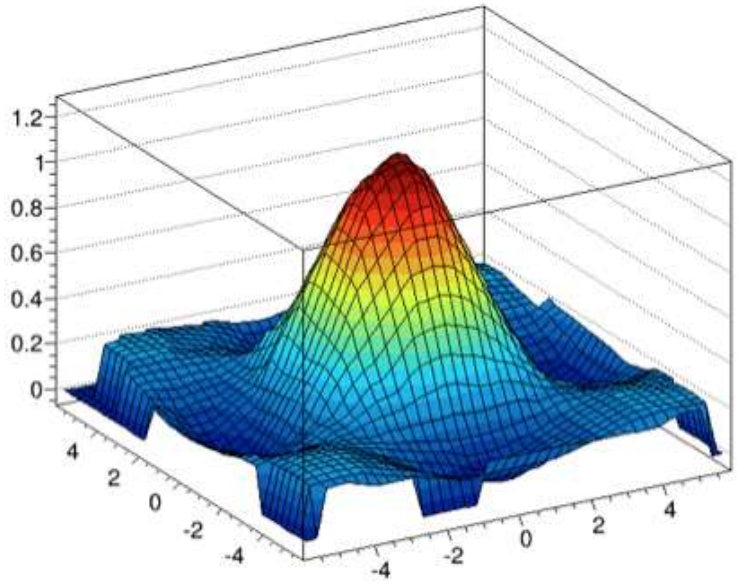
graph.dat 根据指定的格式从ASCII文件“ ”中读取数组。格式的默认值为“ %lg %lg %lg ”。请注意，在最后三种情况中的任何一种情况下，该 SetPoint 方法都可用于更改数据点或添加新数据点。如果数据点index（ i ）大于内部数组的大小，则会自动扩展它们。

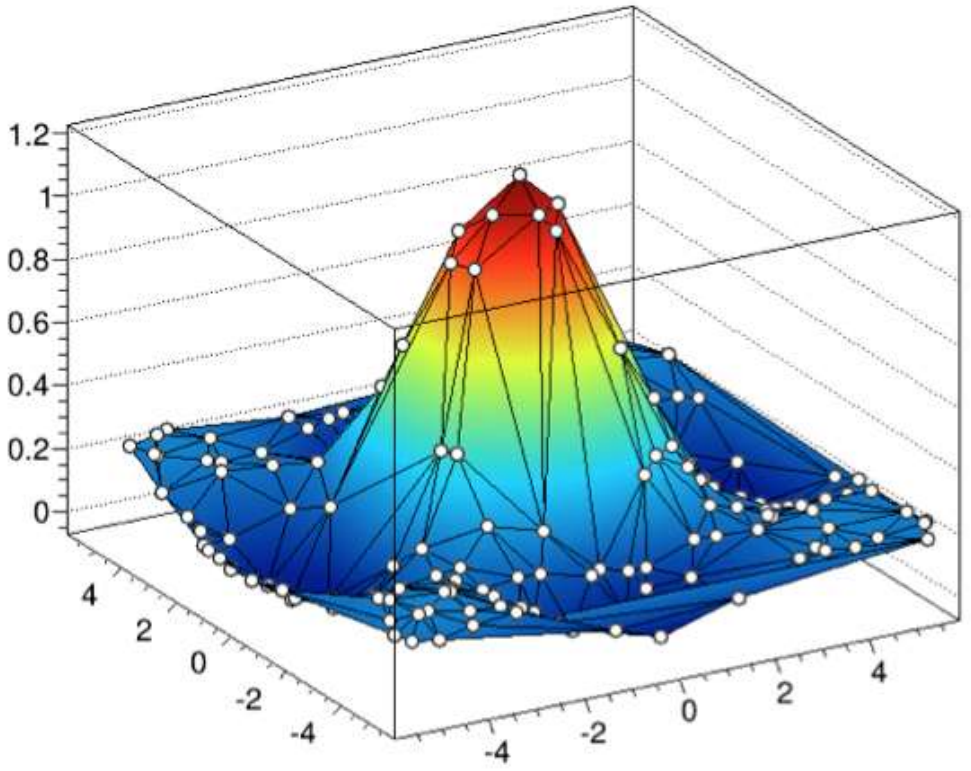
具体的绘图选项可用于绘制 TGraph2D ：

- “ TRI ”使用填充区域绘制Delaunay三角形。使用隐藏表面绘制技术。表面涂有当前填充区域颜色。三角形的边缘涂有当前的线条颜色；

- “ `TRIW` ”Delaunay三角形绘制为线框;
- “ `TRI1` ”Delaunay三角形涂有颜色等级。三角形的边缘涂有当前的线条颜色;
- “ `TRI2` ”Delaunay三角形涂有颜色等级;
- “ `P` ”在每个顶点绘制一个标记;
- “ `P0` ”在每个顶点绘制一个圆圈。每个圆圈背景都是白色的。

`TGraph2D` 也可以使用对2D直方图绘制有效的任何选项绘制A. 在这种情况下，使用Delaunay三角形技术填充中间2D直方图以内插数据集。给定一些现有点的 `TGraph2D` 线性插值 `Z` 任意 `(X,Y)` 点的值 `(X,Y,Z)` 。现有的 `(X,Y,Z)` 点可以随机分散。该算法通过连接现有点来制作Delaunay三角形 `(X,Y)` 。然后用它们来定义平面 `(X,Y,Z)` 插值。因此，插值表面采用以不同角度细分三角形的形式。输出可以采用2D直方图或矢量的形式。找到的三角形可以用3D绘制。无法保证此软件在所有情况下都能正常运行。最初编写的是在 `XY` 具有相似 `X` 和 `Y` 范围的空间中使用几百个点。

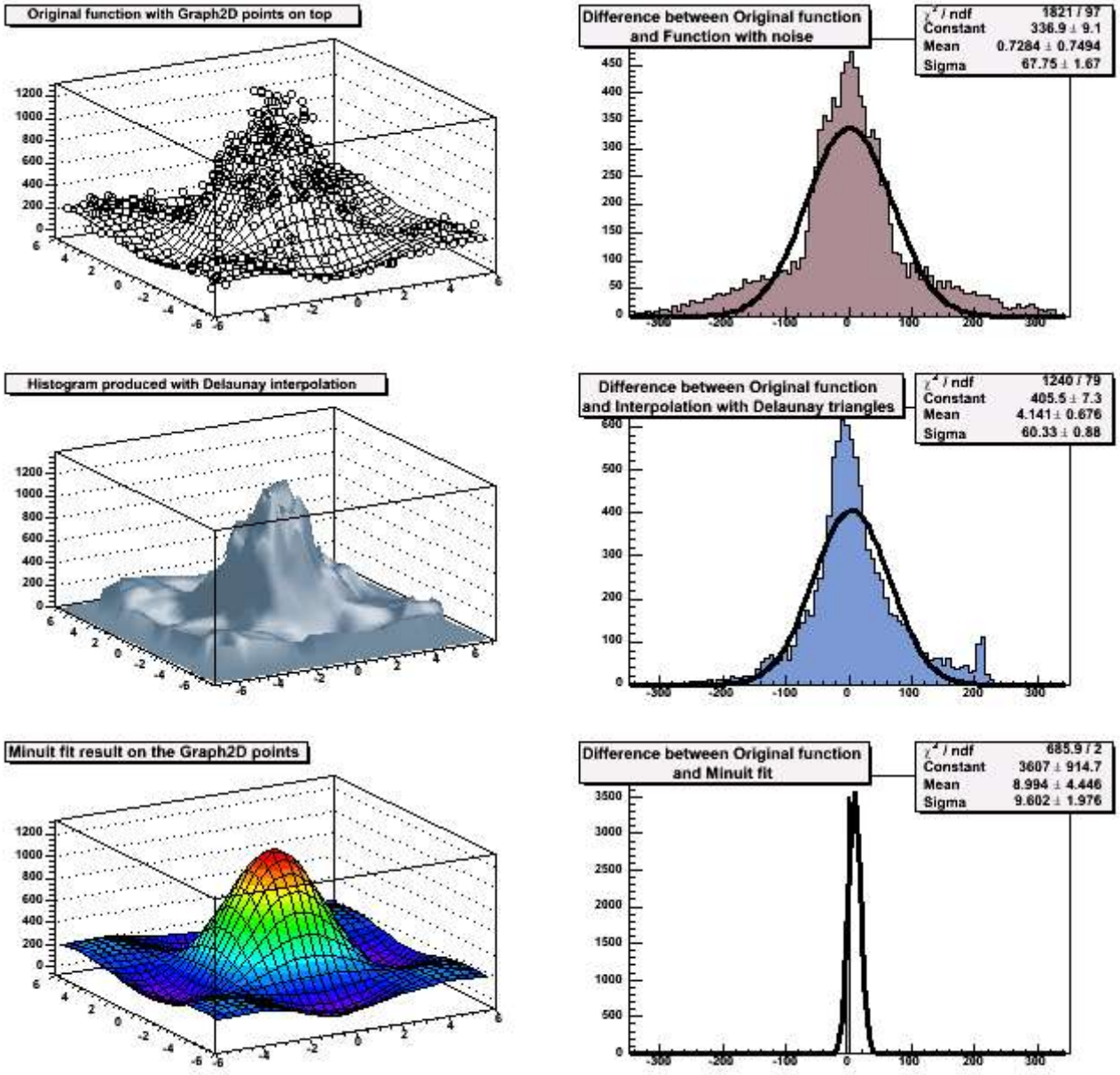




用选项“surf1”和“tri1 p0”绘制的Graph2D

```
{
   TCanvas *c = new TCanvas("c","Graph2D example",0,0,700,600);
   Double_t x, y, z, P = 6.;
   Int_t np = 200;
   TGraph2D *dt = new TGraph2D();
   TRandom *r = new TRandom();

   for (Int_t N=0; N<np; N++) {
      x = 2*P*(r->Rndm(N))-P;
      y = 2*P*(r->Rndm(N))-P;
      z = (sin(x)/x)*(sin(y)/y)+0.2;
      dt->SetPoint(N,x,y,z);
   }
   gStyle->SetPalette(55);
   dt->Draw("surf1");       // use "surf1" to generate the left picture
}                           // use "tri1 p0" to generate the right one
```

一个更完整的例子是 `$ROOTSYS/tutorials/fit/graph2dfit.C` 产生下一个数字。



输出宏graph2dfit.C

6.11 TGraph2DErrors

A `TGraph2DErrors` 是 `TGraph2D` 有错误的。在2D图形上执行错误拟合很有用。一个例子是宏 `$ROOTSYS/tutorials/graphs/graph2derrorsfit.C` 。

6.12拟合图表

图形 `Fit` 方法通常与图像方法相同 `TH1::Fit` 。请参见“拟合直方图”。

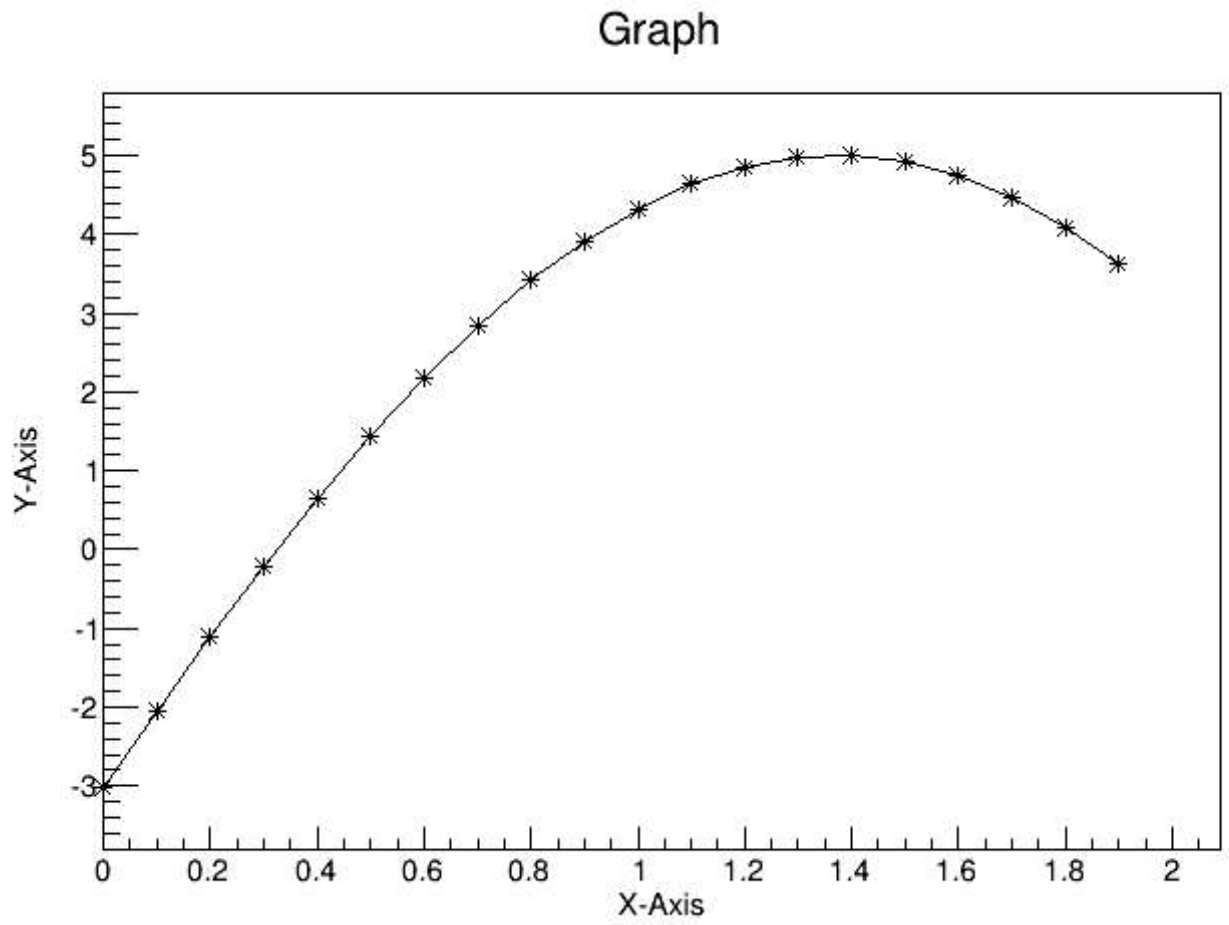
6.13设置图形的轴标题

要为图形的轴提供标题，您需要先绘制图形，然后才能实际拥有轴对象。绘制完成后，通过获取轴并调用 `TAxis::SetTitle` 方法来设置标题，如果要将其居中，则可以调用该 `TAxis::CenterTitle` 方法。

假设 `n, x,` 和 `y` 定义。下一个代码设置 `x` 和 `y` 轴的标题。

```
root[] gr5 = new TGraph(n,x,y)
root[] gr5->Draw()
<TCanvas::MakeDefCanvas>: created default TCanvas with name c1
root[] gr5->Draw("ALP")
root[] gr5->GetXaxis()->SetTitle("X-Axis")
root[] gr5->GetYaxis()->SetTitle("Y-Axis")
root[] gr5->GetXaxis()->CenterTitle()
root[] gr5->GetYaxis()->CenterTitle()
root[] gr5->Draw("ALP")
```

欲了解更多图形示例，请参阅该脚本： `$ROOTSYS/tutorials` 目录 `graph.C` ， `gerrors.C` ， `zdemo.C` ，和 `gerrors2.C` 。



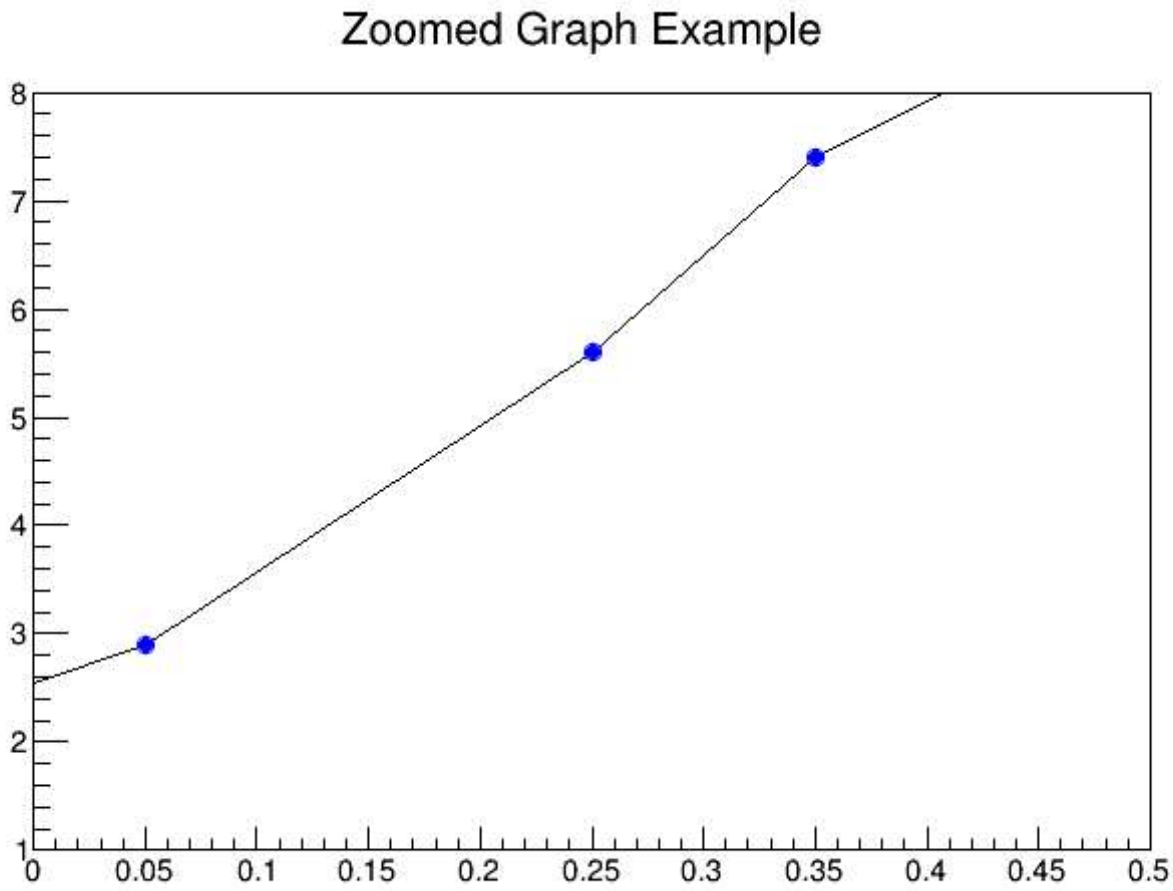
带有轴标题的图形

6.14缩放图表

要缩放图形，您可以先创建具有所需轴范围的直方图。绘制空直方图，然后使用直方图中的现有轴绘制图形。

```
{
   c1 = new TCanvas("c1","A Zoomed Graph",200,10,700,500);
   hpx = new TH2F("hpx","Zoomed Graph Example",10,0,0.5,10,1.0,8.0);
   hpx->SetStats(kFALSE);   // no statistics
   hpx->Draw();
   Int_t n = 10;
   Double_t x[n] = {-.22,.05,.25,.35,.5,.61,.7,.85,.89,.95};
   Double_t y[n] = {1,2.9,5.6,7.4,9,9.6,8.7,6.3,4.5,1};
   gr = new TGraph(n,x,y);
   gr->SetMarkerColor(4);
   gr->SetMarkerStyle(20);
   gr->Draw("LP");// and draw it without an axis
}
```

下一个示例是与上面相同的图形，在x和y方向上放大。



缩放图

6.15图形的用户界面

该类 `TGraphEditor` 提供用于以交互方式设置以下图形属性的用户界面：

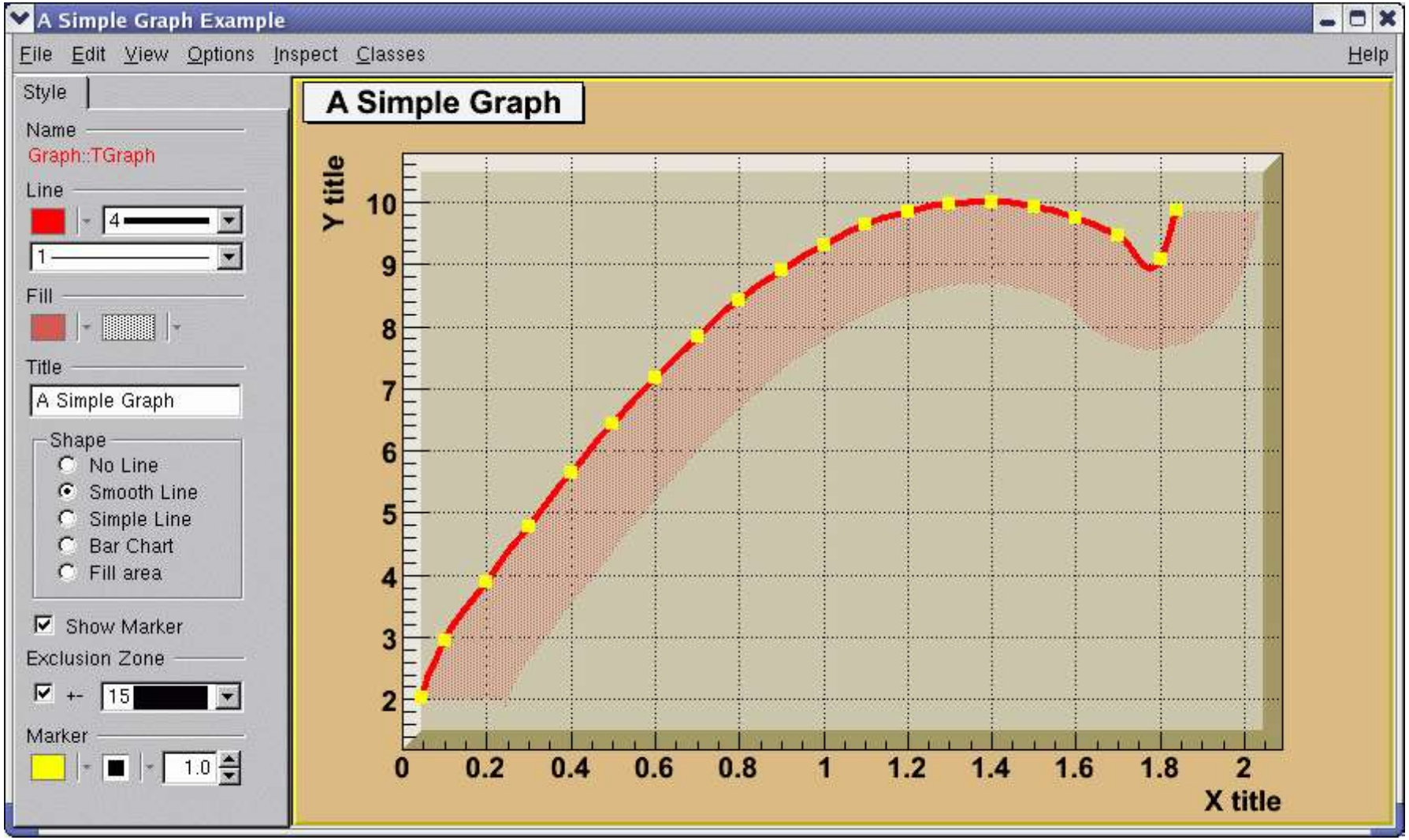
- 标题文本输入字段...设置图表的标题。
- 形状单选按钮组 - 设置图形形状：
  - *无线*：绘制未连接的点；

- *平滑线条*：平滑曲线;
- *简单线*：简单的多线;
- *巴特图*：每个点的条形图。
- *填充区域*：绘制填充区域。

- 显示标记 - 将标记设置为可见或不可见。

- 禁区 - 指定禁区参数：

  - '+ - '*复选按钮*：设置将在线的哪一侧绘制禁区;
  - *宽度组合框*：定义区域的宽度。


A Simple Graph Example
File Edit View Options Inspect Classes Help
Style
Name
Graph::TGraph
Line
4
1
Fill
Title
A Simple Graph
Shape
No Line
Smooth Line
Simple Line
Bar Chart
Fill area
Show Marker
Exclusion Zone
+-
15
Marker
1.0
A Simple Graph
Y title
10
9
8
7
6
5
4
3
2
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
X title