使用 gnuplot 科学作图

Gnuplot 中文教程*

版本 1.0

马欢†

February 9, 2012

目录

前言 4

1 安装 5

2 启动 5

3 数学表达式 6

4 简单函数绘图 6

5 坐标取值范围及刻度 9

6 简单数据绘图 11

7 点线风格 14

8 多组数据绘图 16

9 输出 eps 图片 18

10 输出 pdf 和 png 图片 21

11 Enhanced 文本模式 23

12 插入 LaTeX 公式 26

13 栅格以及方程数值解估算 27

14 第二坐标轴 29

15 Gnuplot 的坐标系统及标签 31

16 箭头 33

17 边框和坐标轴 34

18 图例 36

19 对数坐标 38

20 图像尺寸 40

21 极坐标 41

22 参数方程 43

23 误差条 44

24 拟合 46

25 简单 3D 函数绘图 47

26 数据文件存储格式 50

27 3D 数据曲面绘图及边框 51

28 Pm3d 绘图 53

29 色板（palette）设置 56

30 Image 绘图 57

31 等高线图 59

32 等高线的颜色 61

33 Table 输出 63

34 多图（multiplot） 64

35 曲线色彩填充 66

36 填充风格 68

37 柱状图 70

38 阶梯图 73

39 数据平滑 74

40 统计直方图 76

41 三元算符和分段函数 78

42 几何图形对象 80

43 地图及圆圈数据图 81

44 for 循环 83

45 动画和 reread 循环 84

A 附录：互联网资源 86

索引 87

前言

对于科技工作者来说，gnuplot 是一个非常好用的绘图软件。因为感慨于 gnuplot 中文资料和文档的缺乏，我于数月之前在科学网博客开始撰写“谈谈gnuplot”系列博文，至今已写了四十五篇。虽然还有不少细节问题没有涉及，但是我觉得这些博文已经可以构成一个比较完整的 gnuplot 中文教程。对大多数用户来说，这一系列博文已经涵盖了日常应用的需要。为了方便大家阅读，我现在将这一系列博文集结编辑成为此教程，以 CC BY-NC-SA 知识共享协议发布。

如果有朋友对这一系列文章有什么建议，欢迎提出来，我们可以不断完善和补充。希望我的这些文章对推广开源软件和减少盗版软件的使用能有所帮助。

1 安装

gnuplot 是一个小巧实用的数据处理工具，主要用来绘制2D/3D的数据或者函数图像，但是也包含数学计算、拟合等功能。虽然它的名字里有个 “gnu”，但是它和大名鼎鼎的 GNU 没什么关系，使用的授权协议也不是 GNU GPL，所以这里的 “gnu” 是小写，全名应该读作 “new plot”。

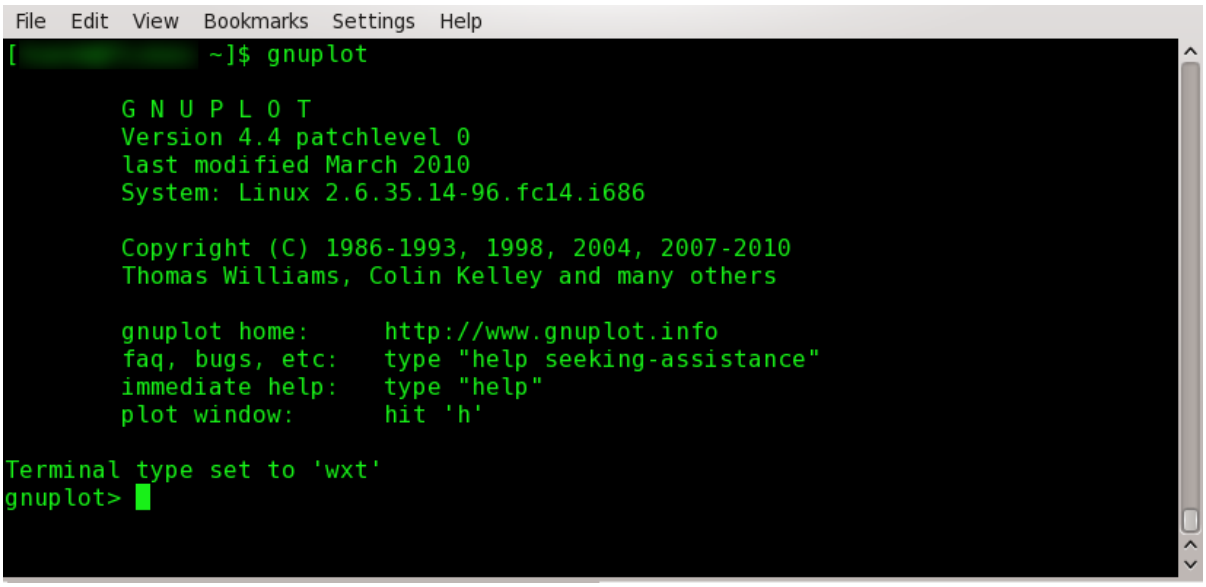
所有的主流 Linux 发行版都包含 gnuplot，因此在 Linux 上安装很简单，只要用各相应发行版的软件安装工具直接安装就可以了。

在苹果电脑上的安装也不复杂。MacPorts 项目把很多 Linux上 的应用程序移植到了苹果系统上，其中就包括 gnuplot。只需要在苹果电脑上安装 MacPorts，之后就能通过 port 命令安装包括 gnuplot 在内的各种 Linux 应用程序了。

在 Windows 下，可以直接到 gnuplot 在 sourceforge 的下载网页下载最新版本（文件名包含 win32 的那个），解压之后到 binary 目录里找到 gnuplot.exe 直接执行就可以了。

2 启动

gnuplot是基于命令行的交互式绘图软件。打开一个终端，输入 gnuplot，随着程序启动，会出现下面的信息：（如果是在 Windows 电脑上，双击 gnuplot.exe 后会自动打开一个命令行窗口）



这里包含 gnuplot 的版本、系统、版权等信息。最关键的是最后一条：

```
Terminal type set to 'wxt'
```

什么是 terminal？在 gnuplot 中，terminal 就是说你打算用什么方式输出图片。这里默认的 terminal 是 wxt，也就是直接输出到电脑屏幕上。gnuplot 支持七十多种 terminal，没必要都记住。我们以后会挑主要的几种讲一下。

提示信息之后，是 gnuplot 的提示符：

```
gnuplot>
```

在提示符之后输入各种命令，就可以开始画图了。

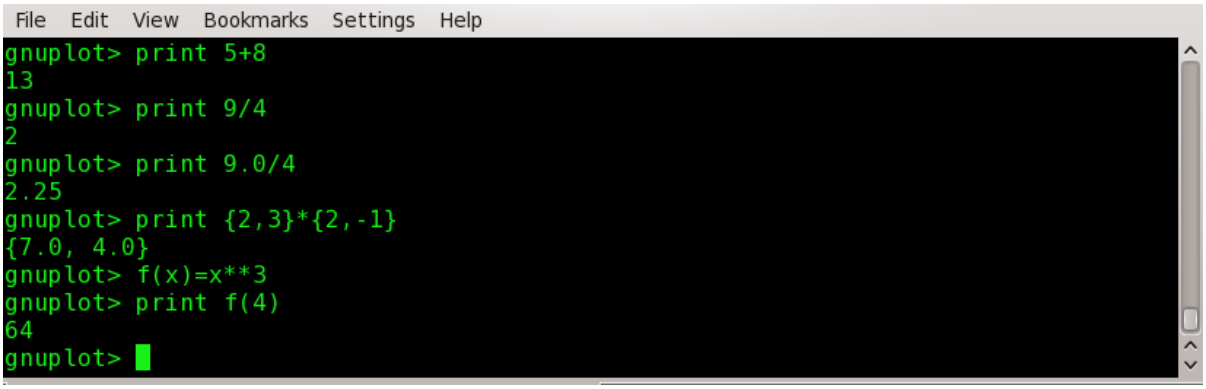
如果要退出程序，只需要输入 quit 或者 exit 命令。

3 数学表达式

在我们开始画图之前，需要知道 gnuplot 里面是如何表达数学公式的。

- **加、减、乘、除、乘方**
  分别用 +，-，*，/，** 表示
- **整数和浮点数**
  和 C 语言类似，gnuplot 对整数和浮点数（实数）区别对待，整数的运算结果还是整数。所以在处理整数除法时要尤其小心，例如 7/2 的结果是 3 而不是 3.5
- **复数**
  gnuplot 支持复数运算，复数用包含在花括号内的一对实数表示，例如{3,5}表示 $3+5i$
- **数学函数**
  gnuplot 含有丰富的数学函数，格式和 C 语言几乎相同。对于实数和复数，函数名是一样的。下面的链接可以看到预定义的函数列表:
  http://www.gnuplot.info/docs_4.2/gnuplot.html#x1-5300013.1
- **自定义函数**
  自定义函数很容易，例如 f(x)=x+1 定义一个一元函数，f(x,y)=x+y 定义一个二元函数。
- π（**圆周率**）
  π 在 gnuplot 里用 pi 表示。

这里是一些例子:



这里用到了 print 命令，就是把结果输出到屏幕上。

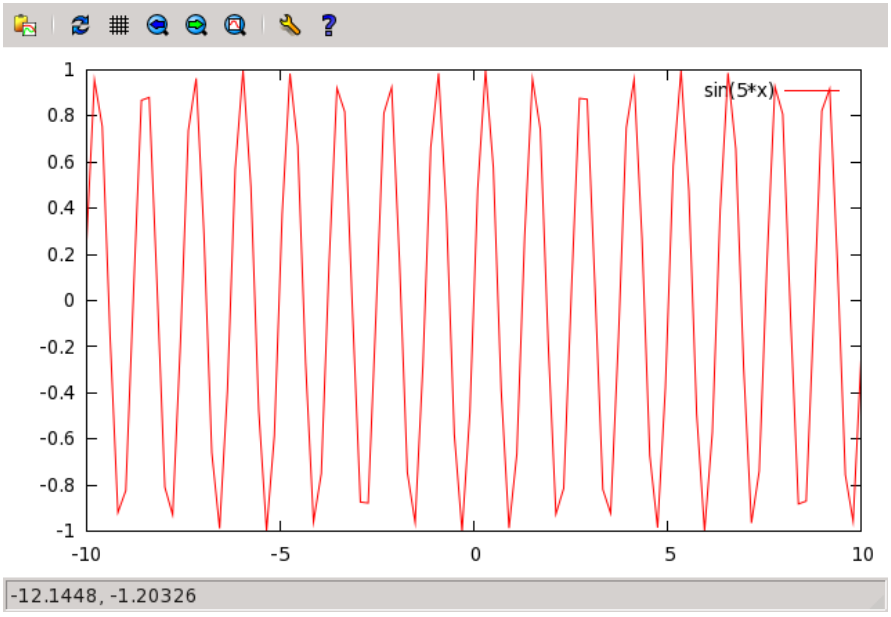
有了这些知识做准备，我们就可以正式开始画图了。

4 简单函数绘图

终于可以开始画图了！先从简单的函数图像入手吧。

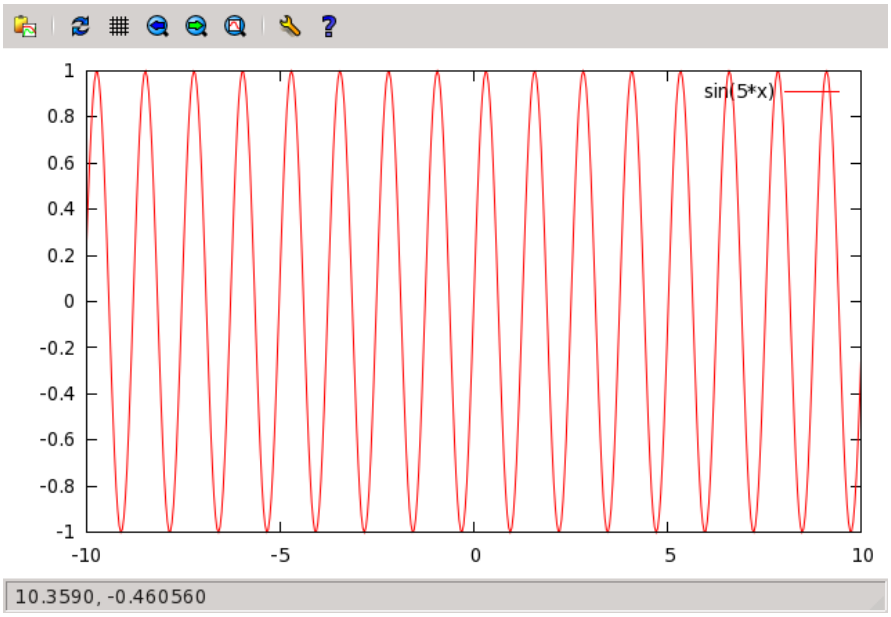
gnuplot 里面的 2D 作图命令是 plot。先试着画一个正弦函数:

```
gnuplot> plot sin(5*x)
```



这个正弦函数看着有点别扭，是吧？这是因为 gnuplot 默认的函数取样为 100 个点，对于快速振荡的函数，这个取样率有点低。函数取样数目由 samples 这个参数控制。在 gnuplot 里面，所有参数赋值都由 set 命令完成。我们把函数取样数目改为 500：

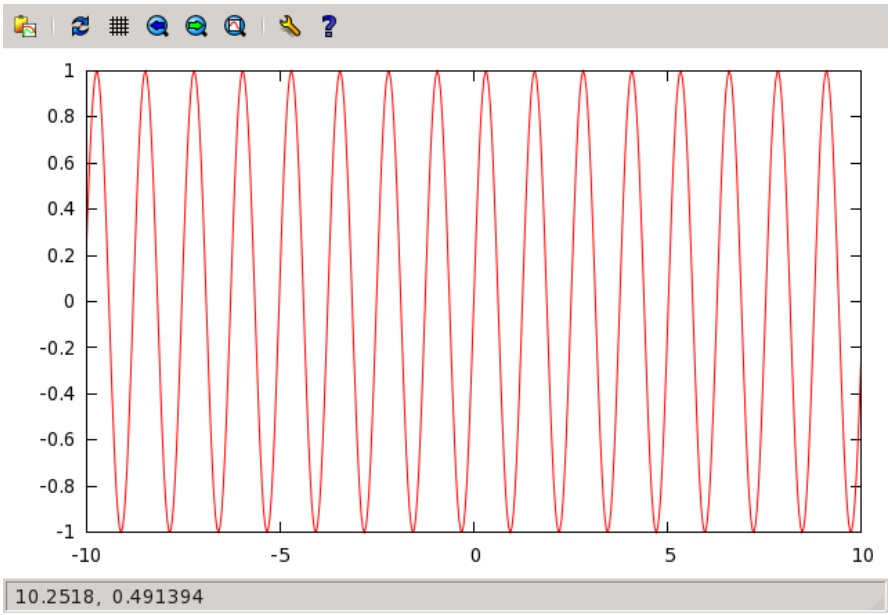
```
gnuplot> set samples 500
gnuplot> replot
```



怎么样，现在看起来好多了吧？这里用到了 replot 命令。顾名思义，replot 就是把刚才的 plot 命令重新执行一遍。

图像右上角的图例（就是那个 sin(5*x) 后面带一段横线）看起来有点碍眼。在 gnuplot 里面，这个图例叫做 key。对于同时包含多组数据的图像来说，图例是必要的。但是现在图像只包含一个函数，而且这个图例的位置也不那么对劲儿，我们先把它去掉：

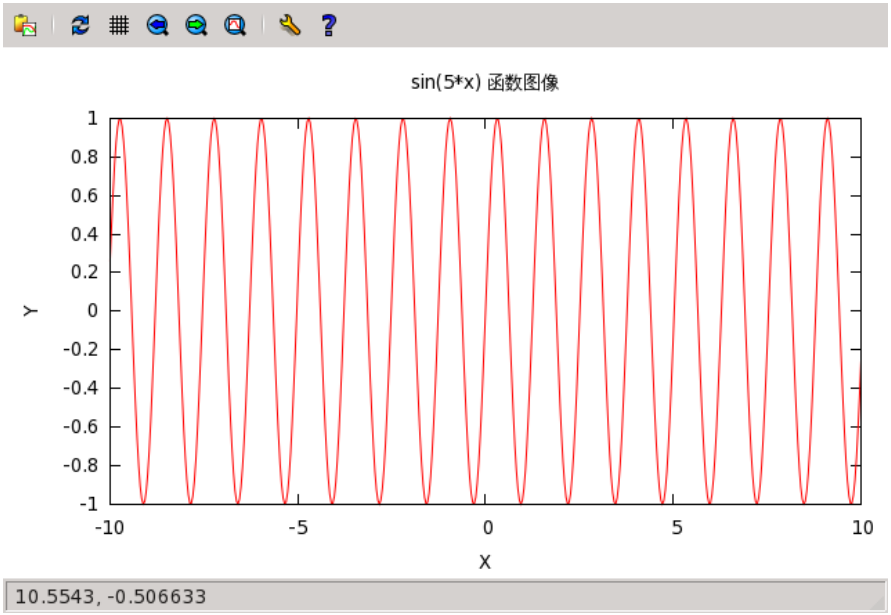
```
gnuplot> unset key
gnuplot> replot
```



这里我们看到，可以用 unset 命令取消一个参数设置。

现在碍眼的图例没有了，但是随之而来的问题是，我们不知道这个图像究竟表示什么意思。为了让它成为一个完整的科学作图，我们给它加上标题和坐标轴标签：

```
gnuplot> set title "sin(5*x)␣函数图像"
gnuplot> set xlabel "X"
gnuplot> set ylabel "Y"
gnuplot> replot
```



xlabel 和 ylabel 分别表示横轴和纵轴的标签。在 gnuplot 里，很多跟坐标有关的参数，都有相应的 x 和 y 版本。title 虽然可以给图像加上标题，但是在真正的科学论文里意义不大，因为所有论文插图都要求配有文字说明（Caption）。引号内的内容为字符串，大多数情况下双引号和单引号没有区别，除非遇到特殊字符（例如换行符\n），这时候单引号会把特殊字符当成一般字符处理，而双引号会按照特殊字符的意义将其展开。

这里我们注意到，字符串里也可以包含中文。究竟是否支持中文，和 terminal 的种类有关系。有些 terminal 对 Unicode 支持不那么好，这时候显示中文就不那么容易了。好在我们可以选择那些比较有利的 terminal。

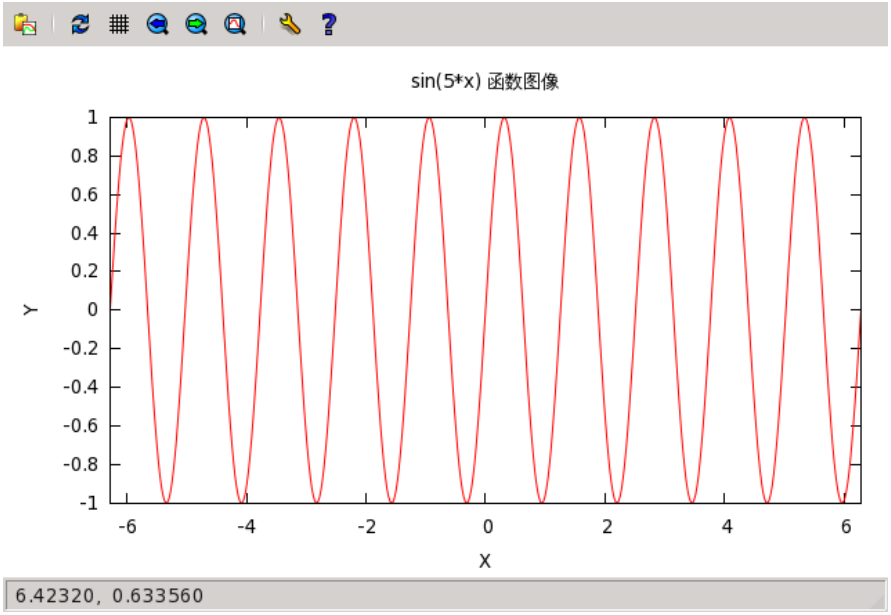
现在我们有了一个简单的函数图像了。虽然看起来还不尽人意，但是没关系，我们以后会把它逐渐完善。

5 坐标取值范围及刻度

我们从上一讲结束时的图像开始。

这里默认的 x 取值范围是从 -10 到 10。我们现在希望 x 的取值范围从 -2π 到 2π，这样函数图像可以正好包括十个周期。横坐标取值范围由 xrange 参数控制。还记得 gnuplot 里面所有参数都由 set 命令控制吗？取值范围由方括号内的一对数表示，两个数之间用冒号隔开：

```
gnuplot> set xrange [-2*pi:2*pi]
gnuplot> replot
```



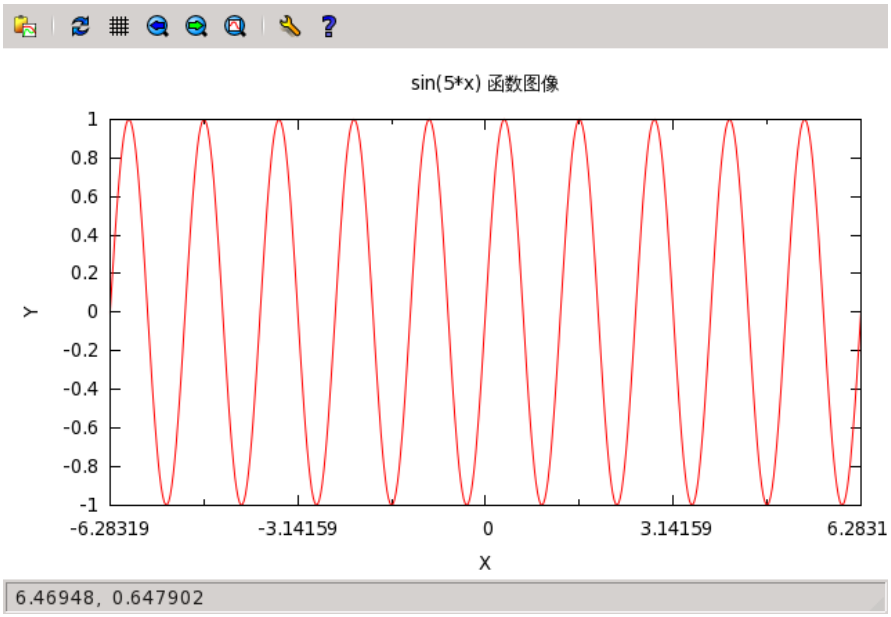
（不要忘记，每一个和 x 坐标有关的参数，都对应有一个和 y 有关的参数，所以纵坐标的取值范围由 yrange 控制。）

现在横坐标取值范围已经符合我们的要求了，但是横轴上的刻度并不是我们想要的。我们不想要刻度出现在整数位置上，而是希望刻度为 π 的整倍数。另外，我们还希望两个主刻度之间 $\pi/2$ 的位置有一个分刻度，分刻度只要有刻度就可以，没必要标出数字了。gnuplot 里面，横轴主刻度和分刻度，分别用 xtics 和 mxtics 表示（m 表示 minor）。我们试试下面的命令：

```
gnuplot> set xtics pi
gnuplot> set mxtics 2
```

```
gnuplot> replot
```

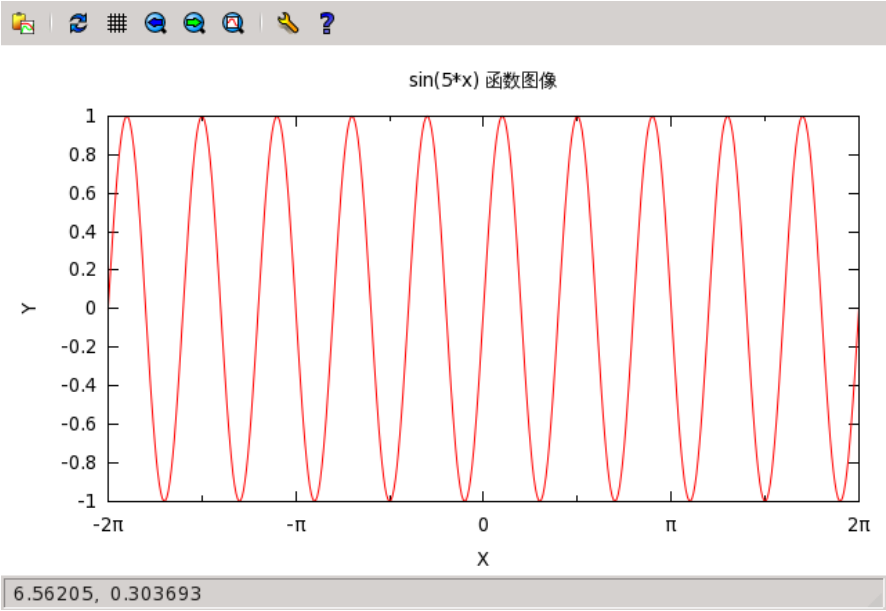
这里的命令表示：横轴主刻度间隔为 π，每两个主刻度之间被分刻度分为 2 份。这组命令得到的图像如下：



现在刻度间隔对了，但是显示的数字并不是我们想要的。我们希望显示字符 π，而不是小数 3.14159。其实，set xtics 命令的形式并不是唯一的。我们试试下面的形式：

```
gnuplot> set xtics ("−2π" −2*pi, "" −1.5*pi 1, "−π" −pi,\
> "" −0.5*pi 1, "0" 0, "" 0.5*pi 1, "π" pi, "" 1.5*pi 1,\
> "2π" 2*pi)
gnuplot> replot
```

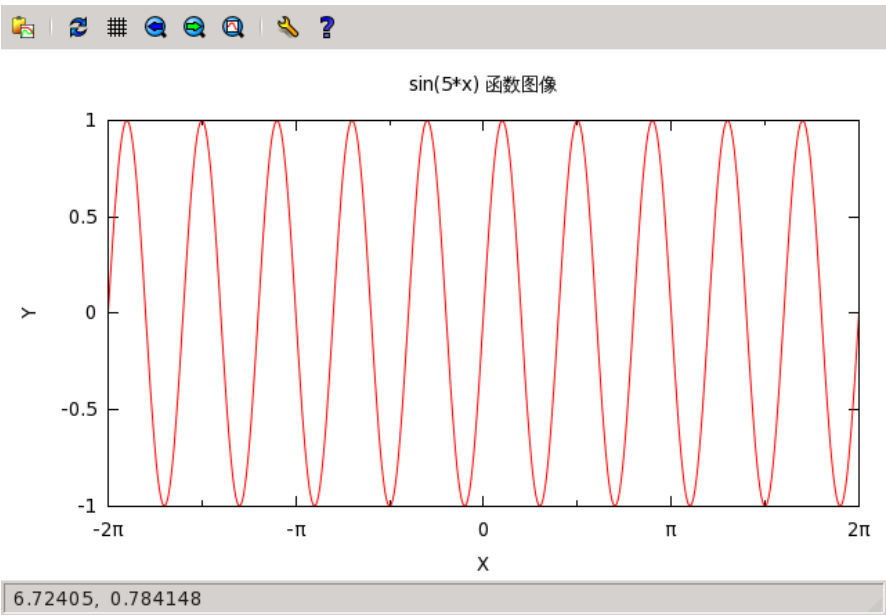
这里 set xtics 命令直接规定了每个刻度的位置和显示的字符。每一个刻度对应三个参数：显示字符、刻度位置、刻度等级。刻度等级为 0 时表示主刻度，等级为 1 时表示分刻度。对于主刻度（等级为 0 时），表示等级的参数也可以省略不写。各个刻度的参数之间用逗号隔开。从上面的例子我们还看出，显示字符可以为空，也就是只标刻度，不显示字符。另外，如果命令太长，需要把一条命令分为多行来写，可以在换行时末尾加上反斜杠（\），表示这条命令还没有结束。这组命令得到的图像如下：



现在横轴坐标已经完全符合我们要求了，我们把纵轴坐标也调整一下，因为我们不需要这么细的划分纵轴刻度：

```
gnuplot> set ytics -1,0.5,1
gnuplot> replot
```

这里又用了 set ytics 命令的另一种形式（再次提醒一下，xtics 和 ytics 语法是完全一样的）：后面跟了逗号隔开的三个参数。这三个参数分别表示：最小主刻度、主刻度步长、最大主刻度。图像如下：



现在我们完成了对于坐标轴的一些微调，图像看起来更顺眼了。

6 简单数据绘图

之前讲过了简单的函数绘图，而实际科研中更多用到的是数据绘图。

在讲如何用 gnuplot 进行数据绘图之前，我们先介绍一个 gnuplot 里最有用的命令：help。

gnuplot 内含一个非常有用的帮助系统。随着我们学习的命令越来越多，记住所有命令的用法不太现实，所以我们会更多的依赖这个帮助系统。例如，我们想知道 set xtics 命令的用法，只要在 gnuplot 里输入：

```
gnuplot> help set xtics
```

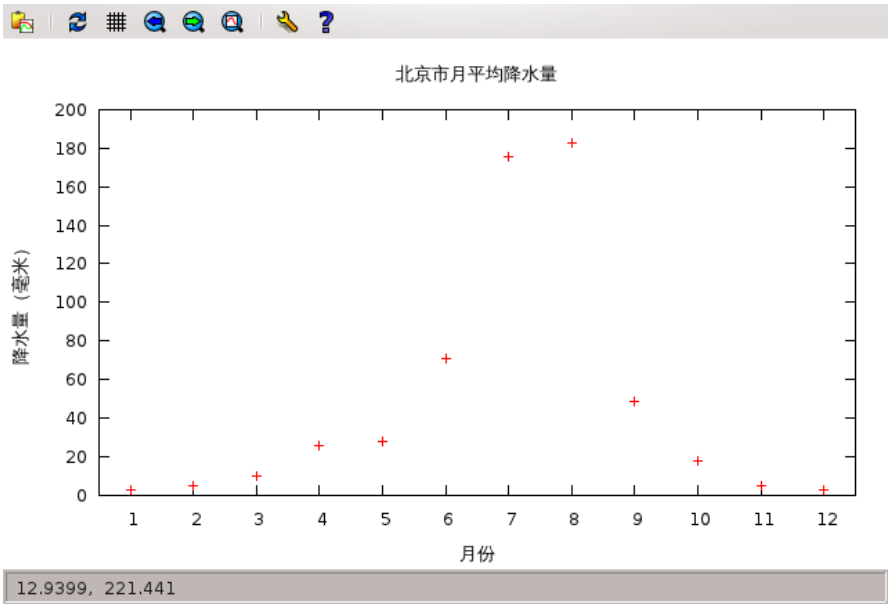
就能得到完整的 set xtics 用法及实例。

好了，下面我们讲数据绘图。首先介绍我们的数据文件。这是一个关于北京市一年中每月平均降水量的数据，我们的数据以纯文本方式储存在文件名为 datafile.dat 的文件中，文件全文如下：

```
### 文件开始 ###
# 北京月平均降水量 (mm)
#
# 月份      降水量
# ==============
1           2.5
2           5.1
3           10.2
4           25.4
5           27.9
6           71.1
7           175.3
8           182.9
9           48.3
10          17.8
11          5.1
12          2.5
### 文件结束 ###
```

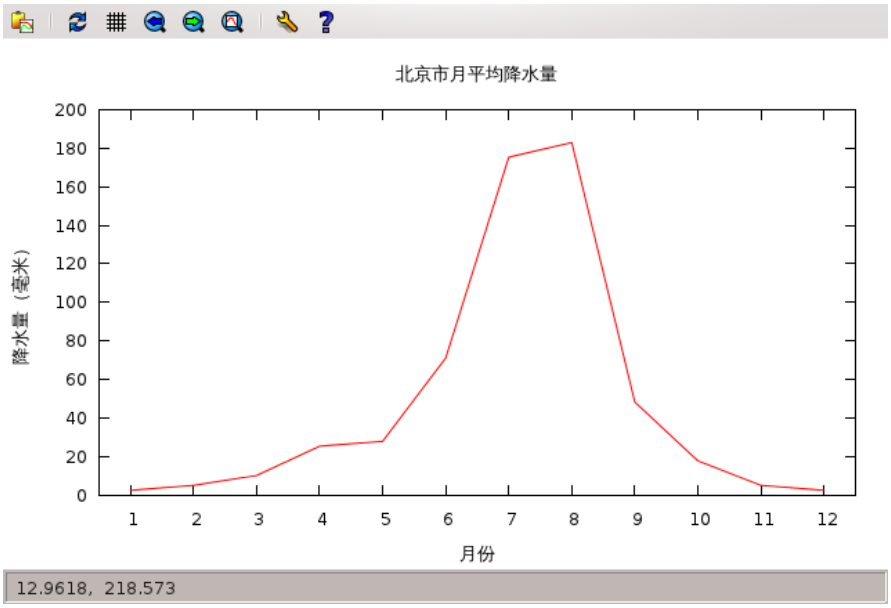
数据分为两列，第一列为月份，第二列为降水量。以 # 符号开始的各行为注释，也就是说，这些行对绘图不构成任何影响。下面开始画图。

```
gnuplot> set xlabel "月份"
gnuplot> set ylabel "降水量（毫米）"
gnuplot> set title "北京市月平均降水量"
gnuplot> unset key
gnuplot> set xrange [0.5:12.5]
gnuplot> set xtics 1,1,12
gnuplot> plot "datafile.dat"
```



这里我们看到，默认的数据作图是画出各数据点。如果我们希望得到数据连线呢？可以使用下面的命令：

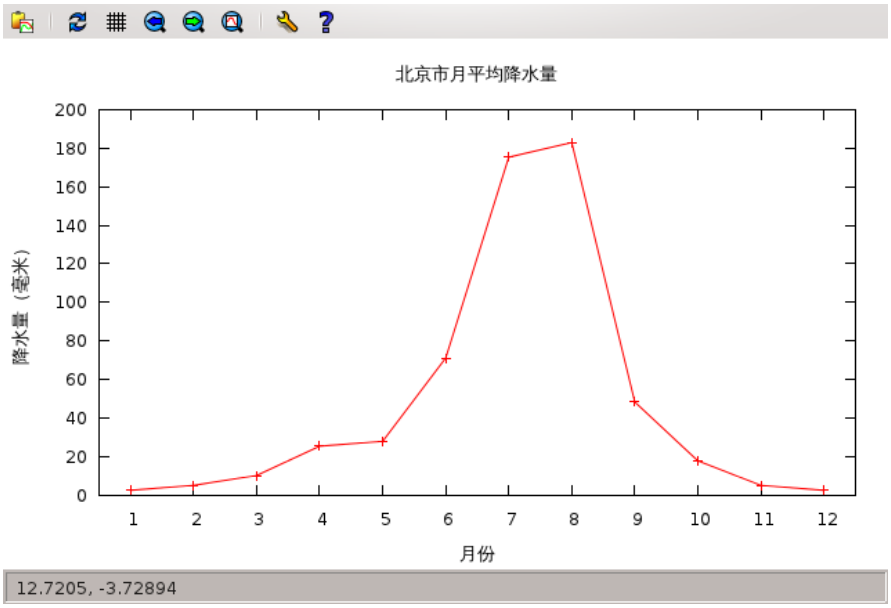
```
gnuplot> plot "datafile.dat" with lines
```



with 命令后面跟的是画图方式，这里使用的是 lines 方式，也就是把各个数据用直线连接起来。gnuplot 大约支持三十种画图方式，默认为 points 方式，我们以后会选择一些常用的方式来介绍。

现在问题来了：如何既得到连线，又得到数据点呢？对了，用 with linespoints：

```
gnuplot> plot "datafile.dat" with linespoints
```



好了，我们已经有一个简单的数据图了。下次我们想办法让它看起来更漂亮一些。

7 点线风格

我们接着上次的数据图谈起。上次我们得到了一个“点线”图，它的数据点是由小“+”字表示的，但是似乎太小了，有点看不清楚。另外，如果我们想在做报告时把这个图用到幻灯片中去，小“+”字很不醒目，这时候我们可能想用其他的标志。gnuplot 里面有几个控制点和线画法风格的参数：

linestyle 连线风格（包括linetype，linewidth等）

linetype 连线种类

linewidth 连线粗细

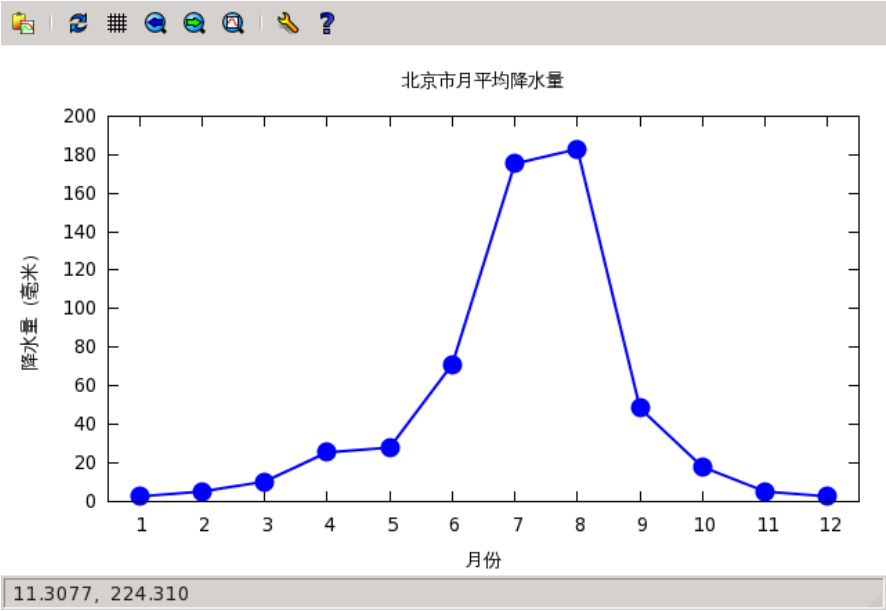
linecolor 连线颜色

pointtype 点的种类

pointsize 点的大小

我们看下面的例子：

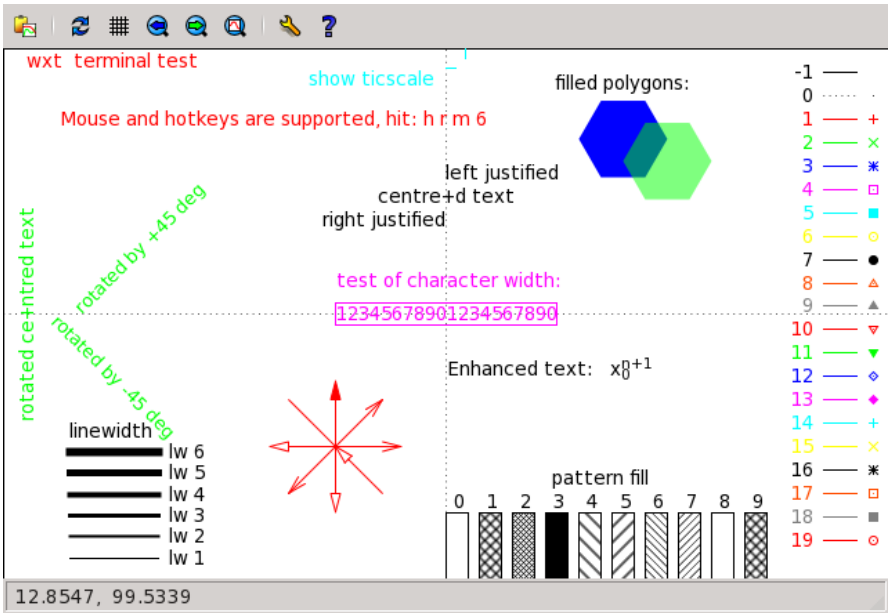
```
gnuplot> plot "datafile.dat" with linespoints linecolor 3 linewidth 2\
> pointtype 7 pointsize 2
```



这几个参数的用法不难理解，直接跟在 with 命令之后就可以了，但是2、3、7这些数字都代表什么意思呢？这些数字是代表不同画法风格的代码，具体某个数字代表什么意思，这个依赖于我们使用的 terminal（还记得我们在第二讲里曾经讲过的 terminal 吗？）拿我们现在正在使用的 wxt terminal 举例，如果想知道这些数字究竟代表什么意思，可以输入命令：

```
gnuplot> test
```

这样当前 terminal 会输出一个测试图：



测试图中包含当前 terminal 的风格代码实例。例如，左下角显示的是连线粗细，右边显示的是色彩和数据点显示风格对应代码。

最后，告诉大家一个好消息：gnuplot 里面很多命令有缩写形式。例如上面例子中的绘图命令可以简写为：

```
gnuplot> plot "datafile.dat" w lp lc 3 lw 2 pt 7 ps 2
```

至于其他参数命令的缩写形式，相信不难猜出来，大家可以试验一下猜猜看。

8 多组数据绘图

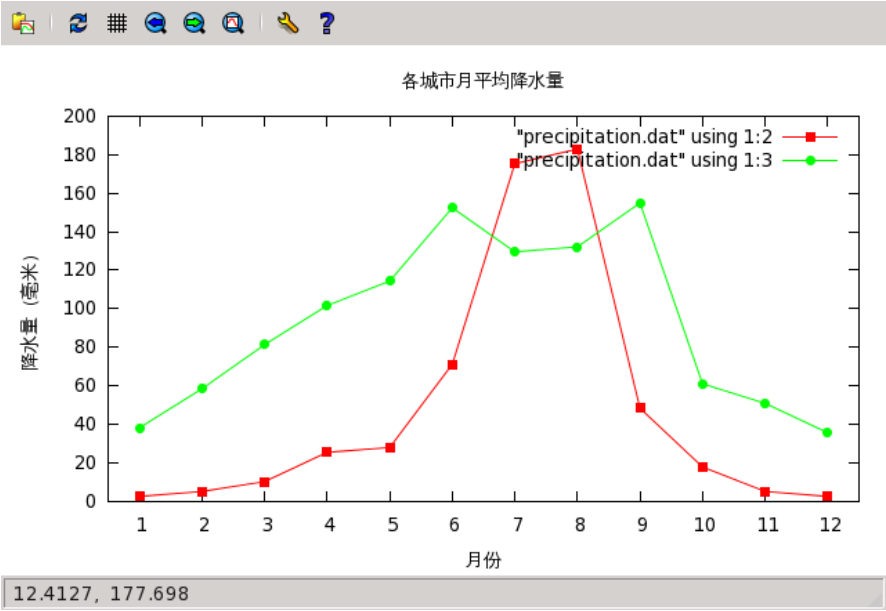
之前讲了如何用 gnuplot 进行数据绘图。实验中经常碰到的情况是，我们要同时处理多组数据。这一次我们来讲一讲，怎样把多组数据绘制到同一个图上。

还拿城市降水量举例，下面是我们的数据文件，文件名是“precipitation.dat”：

```
### 文件开始 ###
# 各城市月平均降水量 (mm)
#
# 月份    北京      上海
# ========================
1         2.5       38.1
2         5.1       58.4
3         10.2      81.3
4         25.4      101.6
5         27.9      114.3
6         71.1      152.4
7         175.3     129.5
8         182.9     132.1
9         48.3      154.9
10        17.8      61.0
11        5.1       50.8
12        2.5       35.6
### 文件结束 ###
```

我们在 gnuplot 里面执行下面的命令：

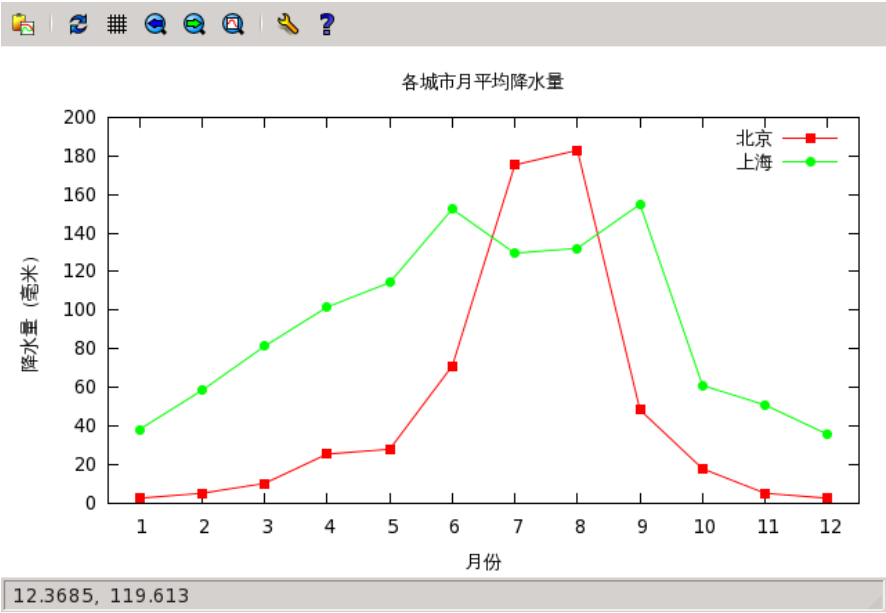
```
gnuplot> set xlabel "月份"
gnuplot> set ylabel "降水量（毫米）"
gnuplot> set title "各城市月平均降水量"
gnuplot> set xrange [0.5:12.5]
gnuplot> set xtics 1,1,12
gnuplot> plot "precipitation.dat" using 1:2 w lp pt 5,\
> "precipitation.dat" using 1:3 w lp pt 7
```



这里我们用了一个新的命令：using。在我们的数据文件包含超过一组数据时，我们可以用 using 指定使用哪列数据。例如 using 1:2 表示使用第一列和第二列数据，第一列为横轴，第二列为纵轴。以此类推，using 1:3 表示使用第一列和第三列数据。如果想把多组数据绘制到一个图上，只要使用一个 plot 命令，后面跟多组数据，每组数据之间用逗号隔开就可以了。

显然，这里的图例又把图像搞乱了。我们没有像以前那样把图例去掉，因为这里有两组数据，我们需要保留图例。怎么办呢？我们来使用下面的命令：

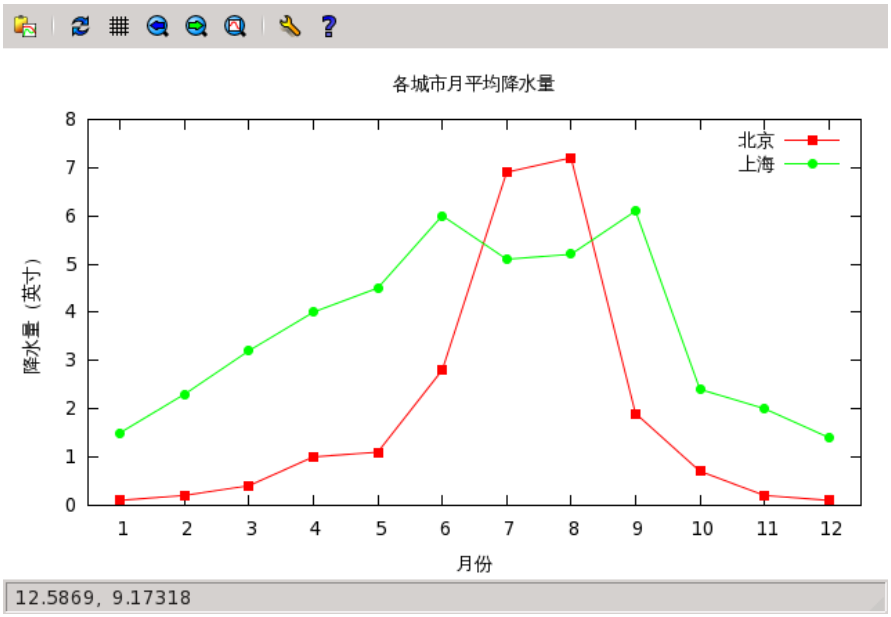
```
gnuplot> plot "precipitation.dat" u 1:2 w lp pt 5 title "北京",\
> "precipitation.dat" u 1:3 w lp pt 7 title "上海"
```



注意到了吗？这里我们使用了字母 u 作为 using 的缩写。另外，这里用了新的参数 title。这里的 title 和之前我们用过的 set title 不同。set title 指定的是整个图的标题，而这里的 title 跟在每一组数据参数后面，指定的是每组数据对应的图例中的 title。这样，我们的图看起来整洁多了。

using 命令除了指定所用的数据列，还可以对数据进行运算操作。例如，我们现在还是绘制上面的数据，但是降水量单位使用英寸而不是毫米。我们知道，1英寸=25.4毫米，所以我们执行下面的命令：

```
gnuplot> set ylabel "降水量（英寸）"
gnuplot> plot "precipitation.dat" u 1:($2/25.4) w lp pt 5 title "北京",\
> "precipitation.dat" u 1:($3/25.4) w lp pt 7 title "上海"
```



在对特定列的数据进行运算操作时，我们需要在列号之前加上 $ 符号，这样表示该数据的值。

虽然这里的例子是同一个文件里的多组数据，但是如果数据存在多个文件里，这里的命令同样适用。

9 输出 eps 图片

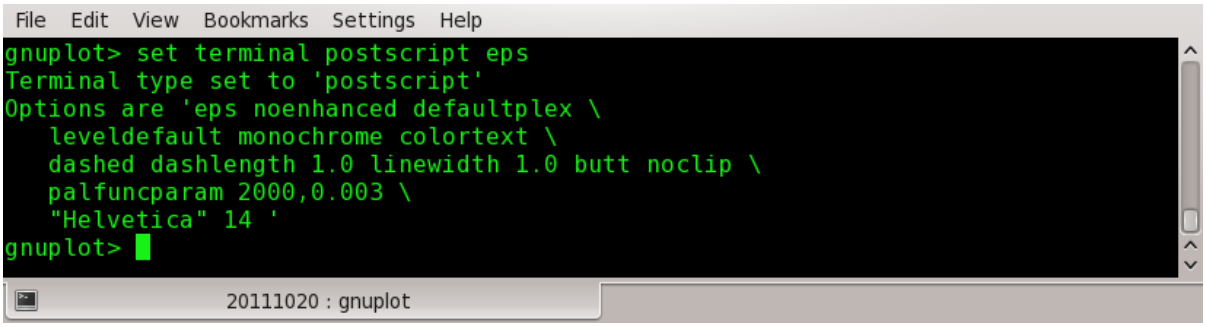
到目前为止，我们所有的 gnuplot 作图都只是输出到电脑屏幕上。如果要在我们的文档里使用这些图片，我们必须把它们以某种文件格式存储下来。前面已经介绍过，gnuplot 里面控制图像输出方式的命令是 terminal。我们这次就讲一下如何把图片输出为 eps 文件。

之所以首先选择 eps，是因为大量科学技术文档使用 LaTeX 来编辑排版，而 eps 是 LaTeX 最常用的图片格式，支持高质量的矢量图形，并且可以方便的转换为 pdf、svg 等其他常用格式。

首先，让我们看看如何设置 terminal。在gnuplot里输入下面的命令：

```
gnuplot> set terminal postscript eps
```

这里的 terminal 其实是 postscript，而 eps 是作为 postscript 的一个参数。这并不奇怪，因为 eps 本来就是 postscript 衍生出来的。输入这个命令之后，gnuplot会自动返回下面的信息：



这里给出了一些默认的其它参数。gnuplot 下的每一个 terminal 所带的参数不完全相同，一些参数只针对某一 terminal。我们只需要知道一些常用 terminal 的常用参数，而其它的 terminal 和参数，我们可以在使用的时候通过 help 命令查询。现在我们暂且不管这些参数，先画一个图试试。还是用我们上次用过的降水量数据文件：

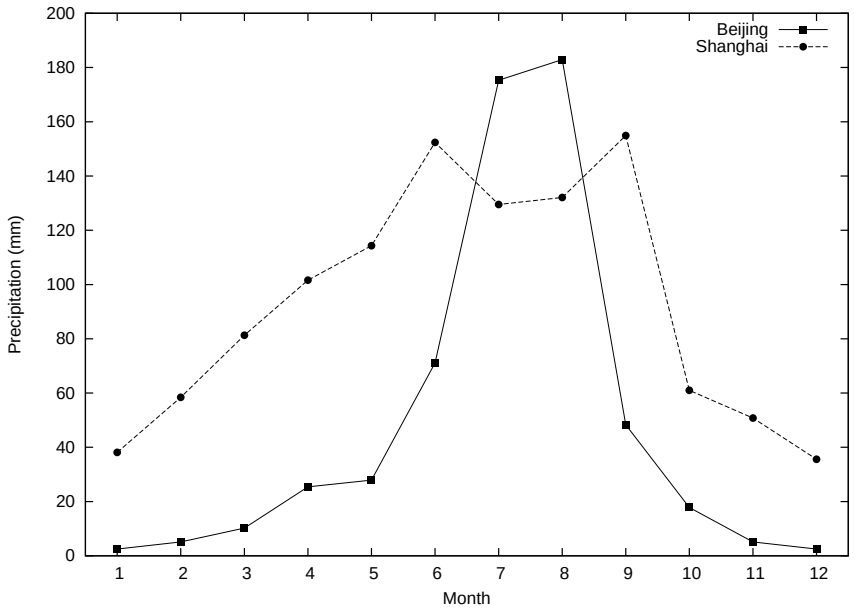
```
gnuplot> set xlabel "Month"
gnuplot> set ylabel "Precipitation (mm)"
gnuplot> set xrange [0.5:12.5]
gnuplot> set xtics 1,1,12
gnuplot> set output "precipitation.eps"
gnuplot> plot "precipitation.dat" using 1:2 w lp pt 5 title "Beijing",\
> "precipitation.dat" using 1:3 w lp pt 7 title "Shanghai"
gnuplot> set output
gnuplot> set term wxt
```

这里我们没有使用中文标签，因为在 postscript 使用中文字体并不容易。以后我们会介绍如何绕过这个障碍。

第五行有一个新命令：set output。这没什么多说的，就是指定输出文件的文件名。第七行还有一个 set output，但是后面没有跟任何文件名。这其实是告诉 gnuplot，这个文件已经输出完毕，可以关闭了。另外也可以用 unset output，其实是相同作用。之所以这么做，是因为有些 terminal 支持多页文件，所以在画完图之后，gnuplot 并不自动关闭文件，等待你输入下一页内容，除非你手动关闭，切换 terminal，或者退出 gnuplot 程序。

最后，别忘了把 terminal 切换回屏幕显示，这样可以避免一些意想不到的事情发生。注意：我们又用缩写了！

下面让我们来看看生成的 eps 文件吧：



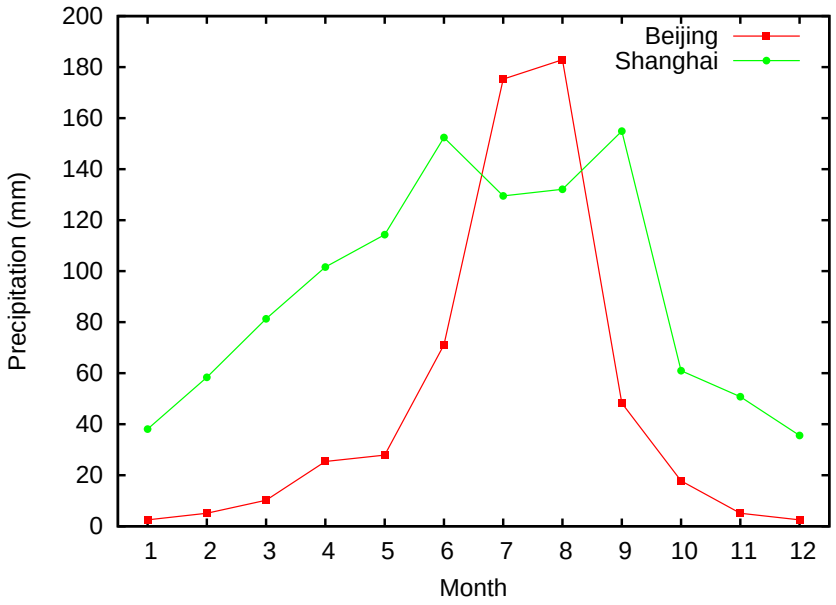
这个图像可能并不是我们想要的。有几个地方我们想要修改一下：

- 这个图是黑白的！我们想要彩色图。
- 有一组连线是虚线。这也难怪，黑白图如果不用虚线，还真分辨不出来。但我们想要彩色实线。
- 让直线稍微加粗一点。
- 标签文字显得小了些，我们想要大一点的字体。

我们来重新画一下这个图：

```
gnuplot> set terminal postscript eps color solid linewidth 2 "Helvetica" 20
gnuplot> set output "precipitation-color.eps"
gnuplot> replot
gnuplot> set output
gnuplot> set term wxt
```

这里我们用 color 参数表示我们要彩色图，solid 表示我们要实线不要虚线，linewidth 参数指定 2 倍线宽，而最后指定使用 Helvetica 20 号字体。让我们来看看新生成的 eps 文件：



这里的字体为 postscript level 2 字体，共有 35 种。除了 Helvetica，常用的还有 Times-Roman。如果硬要多记两个字体名字，就记住 Courier 和 Symbol 吧。Courier 是编程时常用的等宽字体，而 Symbol 字体可以用来显示希腊字母。如果你实在好奇还有哪些其它字体，可以看看下面的网页链接：http://xfig.org/userman/attributes.html#font-panel

10 输出 pdf 和 png 图片

这次讲讲怎样把图片输出为 pdf 和 png 格式。

上次讲过的 eps 文件其实很容易转换为 pdf，为什么我们还专门讲一下 pdf 格式输出呢？因为在 postscript terminal 下很难使用中文字体，而在 pdf 下面就容易多了，而 pdf 文件又很容易转换回 eps，这样就绕开了 eps 中文支持不好的问题。

png 是一种无损压缩位图格式，图形质量优于 jpg 等有损压缩格式，支持透明效果，可以生成非常小体积的文件，适于放在网上交流。通过各种图像处理软件，png 文件也很容易转换为其他位图格式。

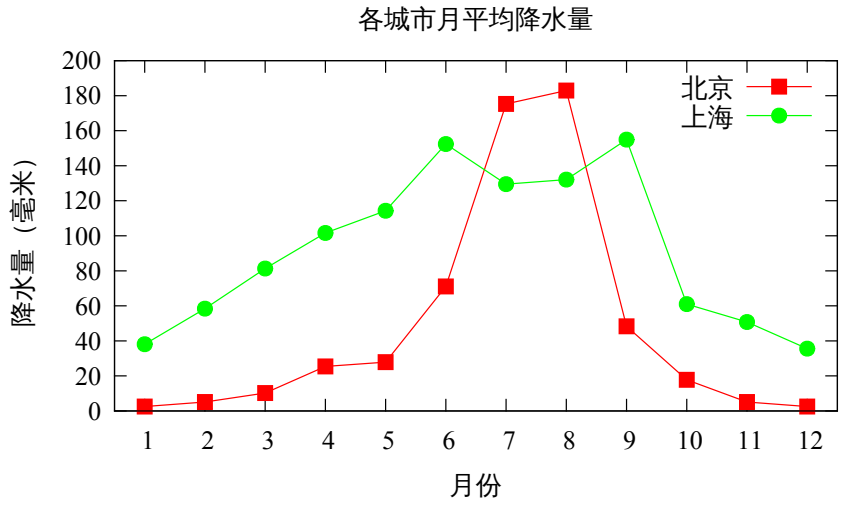
下面首先看一个 pdf 输出的例子，咱们还是用之前用过的城市降水量数据文件：

```
gnuplot> set xlabel "月份"
gnuplot> set ylabel "降水量（毫米）"
gnuplot> set title "各城市月平均降水量"
gnuplot> set xrange [0.5:12.5]
gnuplot> set xtics 1,1,12
gnuplot> set term pdfcairo lw 2 font "Times␣New␣Roman,8"
gnuplot> set output "precipitation.pdf"
gnuplot> plot "precipitation.dat" u 1:2 w lp pt 5 title "北京",\
> "precipitation.dat" u 1:3 w lp pt 7 title "上海"
gnuplot> set output
```

这里我们用的 terminal 是 pdfcairo，而不是简单的 pdf。区别是 pdfcairo 使用了 cairo（一个 2D 图形程序库）和 pango（一个字体渲染程序库）来生成 pdf 文件，优点是更好的国际支持。有了之前的经验，这里的 terminal 参数不需要多解释了。这里我们使用了 “Times New Roman” 8 号字体。

和 eps 下使用 postscript 字体不同，这里可以是电脑系统里安装的任何字体。在 Linux 下，可以使用 fc-list 命令察看系统里到底有哪些字体可用。

下面我们来看生成的 pdf 图片：

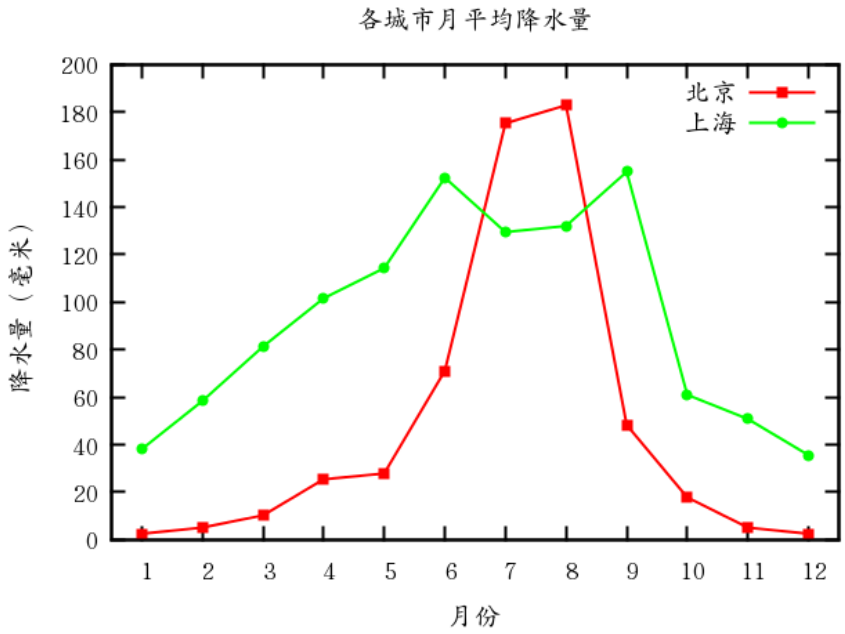


这里有一个小问题：虽然数字使用了 Times New Roman 字体，但是汉字使用了其他字体（这里是我的系统默认的“文泉驿正黑”）。这是因为 Times New Roman 本来就不是中文字体。如果我们想让中英文混排时字体统一，必须使用支持中文的字体。

下面我们来看 png 输出的例子：

```
gnuplot> set term pngcairo lw 2 font "AR␣PL␣UKai␣CN,14"
gnuplot> set output "precipitation.png"
gnuplot> replot
gnuplot> set output
gnuplot> set term wxt
```

基于和上面同样的原因，这里使用的 terminal 是 pngcairo 而不是简单的 png，而字体是 AR PL UKai CN（文鼎简中楷）。下面是生成的 png 图片：



11 Enhanced 文本模式

我们之前的图像里的 title、xlabel 等标签里用到的都是纯文本字符串，如果我们希望输出稍复杂一点的文字标签，例如字母加角标，我们可以使用 enhanced 文本模式。

gnuplot 里面好多 terminal 都支持 enhanced 模式，使用方法就是在 set terminal 的时候，在后面加上 enhanced 参数。enhanced 模式里有一些表达特殊含义的字符，利用这些字符可以构成一些比较复杂的文字输出。这些特殊字符主要包括：

1. ^：表示后面的字符为上角标
2. _：表示后面的字符为下角标
3. @：表示后面的字符不占任何宽度
4. &{"string"}：表示一段空白，空白的长度等于花括号内那段字符串所占宽度
5. ~：表示后面的两个字符重叠打印（相当于打字机在同一位置打印两个字符）；也可以在第二个字符前加上一个数字，表示第二个字符相对于第一个字符有一个竖直方向的移动，移动距离等于该数字乘以字符尺寸。

上面提到的“字符”，也可以是包含在花括号（{}）内的“字符串”。除了上面这些特殊标志，还可以临时改变字符（或字符串）的字体，方法是：

```
{/字体名=字体大小 字符}
```

下面我们看一些例子：

- `"a^x"`
  a^x
- `"a_x"`
  a_x

- `"a^b_{cd}"`
  a^{b}_{cd}
  这里的上下角标没有对齐，因为上标 b 需要占一定宽度

- `"a^@b_{cd}"`
  a^{b}_{cd}
  这里我们看到了 @ 的作用

- `"abc&{de}fg"`
  abc fg
  这里 & 后面的字符没有显示，留出一段和这些字符同样宽度的空白

- `"~a{1.2\\_}"`
  $\bar{a}$
  注意这里的两个反斜杠。因为下横杠（_）是一个特殊字符，需要在前面加反斜杠 \ 来表示它本来的意义，而反斜杠本身也是一个特殊字符，需要在它前面再加一个反斜杠。我们在第四讲里谈到过双引号和单引号的区别。如果我们在这里用单引号而不是双引号，那么只需要一个反斜杠就可以了。这里的 1.2 表示后面的字符（_）向上移动 1.2 个字符大小的距离。

- `{/Symbol abc}abc`
  αβχabc
  这里第一个花括号里的 abc 使用了 Symbol 字体，而后面括号外的 abc 使用的是默认字体。注意字体名称前的斜杠方向。这里我们看到，Symbol 字体里的英文字母对应显示的是希腊字母。Symbol 字体里还包括了一些其它符号，下面是完整的 Symbol 字体列表：

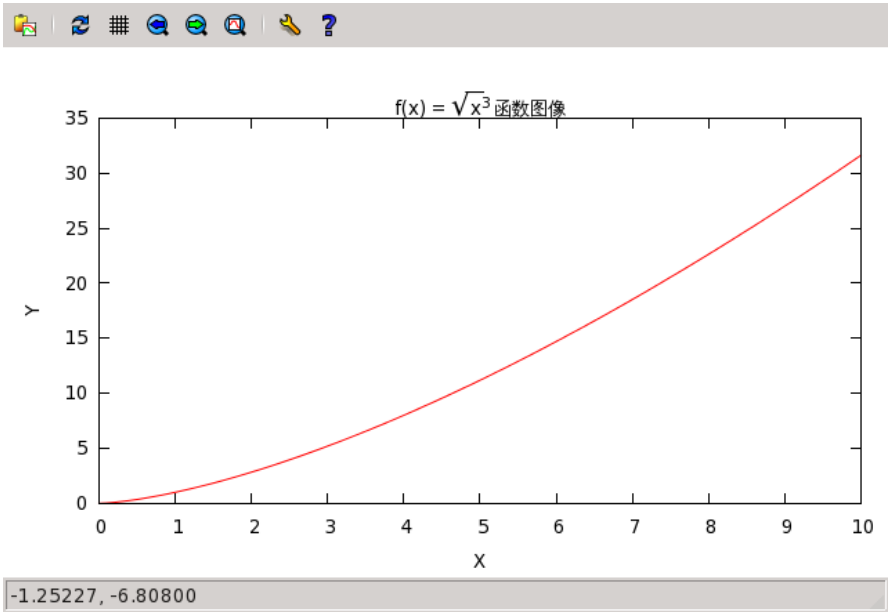
Symbol

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
20		!	∀	#	∃	%	&	∋	(	)	*	+	,	-	.	/
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	≅	Α	Β	Χ	Δ	Ε	Φ	Γ	Η	Ι	ϑ	Κ	Λ	Μ	Ν	Ο
50	Π	Θ	Ρ	Σ	Τ	Υ	ς	Ω	Ξ	Ψ	Ζ	[	∴	]	⊥	_
60	‾	α	β	χ	δ	ε	φ	γ	η	ι	ϕ	κ	λ	μ	ν	ο
70	π	θ	ρ	σ	τ	υ	ϖ	ω	ξ	ψ	ζ	{	|	}	~	
80	*unused*															
90																
A0	€	ϒ	′	≤	⁄	∞	ƒ	♣	♦	♥	♠	↔	←	↑	→	↓
B0	°	±	″	≥	×	∝	∂	•	÷	≠	≡	≈	…	|	–	↵
C0	ℵ	ℑ	ℜ	℘	⊗	⊕	∅	∩	∪	⊃	⊇	⊄	⊂	⊆	∈	∉
D0	∠	∇	®	©	™	∏	√	·	¬	∧	∨	⇔	⇐	⇑	⇒	⇓
E0	◊	〈	®	©	™	∑	⎛	⎜	⎝	⎡	⎢	⎣	⎧	⎨	⎩	⎪
F0		〉	∫	⌠	⎮	⌡	⎞	⎟	⎠	⎤	⎥	⎦	⎫	⎬	⎭	

上面图中显示的是 Symbol 字体对应的16进制代码。例如上面例子中的希腊字母 $\alpha\beta\chi$，对应的16进制代码分别为 61、62、63，转换为8进制代码就是 141、142、143。在 gnuplot 里，我们可以直接用字符的8进制代码表示这个字符，所以我们也可以用 `{/Symbol \141\142\143}` 来表示 $\alpha\beta\chi$。

根据上面的内容，下面是一个综合起来的例子：

```
gnuplot> set term wxt enhanced
gnuplot> set xlabel "X"
gnuplot> set ylabel "Y"
gnuplot> set xrange [0:10]
gnuplot> set xtics 0,1,10
gnuplot> unset key
gnuplot> set title "f(x)␣=␣{/Symbol=16␣\326}~{x^@3}{1.1{/Symbol=16␣\276}}&{aa}\
>␣函数图像"
gnuplot> plot sqrt(x**3)
```



虽然利用 enhanced 模式也能显示一些简单的数学表达式，但是对于稍微复杂一点的公式来说，显示效果无法令人满意。要在 gnuplot 里显示数学公式，终极方案还是要用 LaTeX。

12 插入 LaTeX 公式

上一次我们谈过，在 gnuplot 里使用 enhanced 模式虽然可以生成一些简单的数学表达式，但是对于稍复杂的数学公式来说，enhanced 模式没办法生成令人满意的结果。这里我们介绍 gnuplot 的另外一个 terminal：epslatex。

epslatex 和我们之前介绍过的 postscript eps 输出方式非常接近，因此它们很多参数都是相同的。区别在于，epslatex 使用 postscript eps 仅生成图形存于 eps 文件，而所有文字标签包含在另外一个 LaTeX 文件中。在 gnuplot 完成输出之后，使用 LaTeX 命令最终生成完整的图片。这种做法的好处是不言而喻的，即使在输出完成后，我们仍然可以编辑 LaTeX 文件获得我们想要的显示效果。

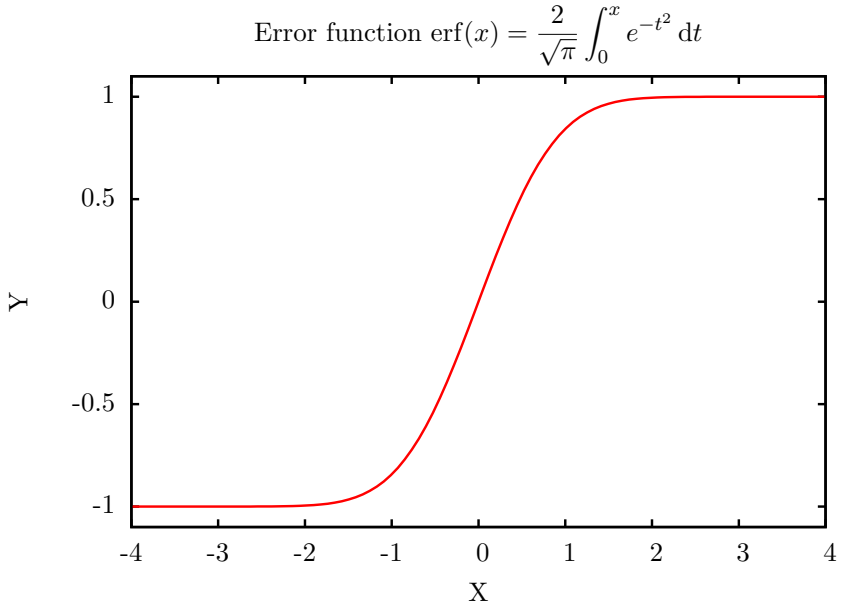
下面我们看例子：

```
gnuplot> set xlabel 'X'
gnuplot> set ylabel 'Y'
gnuplot> set title 'Error␣function␣$\displaystyle\mathrm{erf}(x)␣=␣\frac{2}{\
    sqrt{\pi}}␣\int_0^x␣e^{-t^2}\,\mathrm{d}t$'
gnuplot> set xrange [-4:4]
gnuplot> set yrange [-1.1:1.1]
gnuplot> unset key
gnuplot> set term epslatex standalone lw 2 color 11
gnuplot> set output "erf.tex"
gnuplot> plot erf(x) lw 2
gnuplot> set output
```

前三行的标签文字，我们使用了单引号，避免了双引号内需要连着两个反斜杠的麻烦。在 title 里面，我们使用了 LaTeX 数学公式。在 set term 命令里，standalone 是一个新的参数，它表示生成完整的 LaTeX 文件。如果没有这个参数，生成的 LaTeX 文件将不能单独编译，必须把代码插入其它的 LaTeX 文件中编译。set term 最后的数字 11 代表字体大小。set output 只需指定 LaTeX 文件名，而

不需要指定 eps 文件名。我们谈到过 gnuplot 包含很多预定义的函数，这里的 erf 就是其中之一，表示误差函数。

我们通过 epstopdf 命令将生成的 eps 图片转为 pdf，然后用 pdflatex 命令可以把图片直接编译为 pdf 文件，下面是显示的效果：

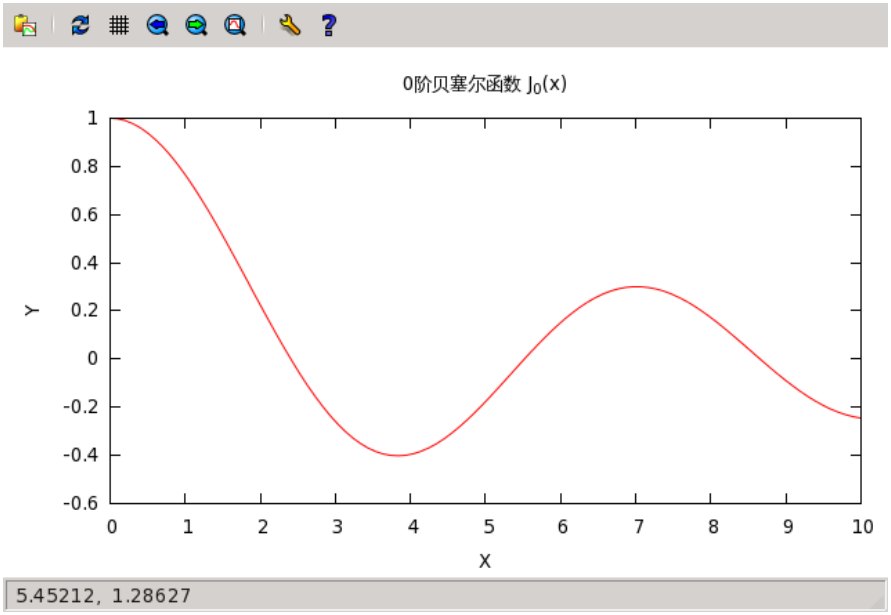


怎么样？LaTeX 的数学公式效果真不是盖的。

13 栅格以及方程数值解估算

我们现在来画一个 0 阶贝塞尔函数 $J_0(x)$：

```
gnuplot> set term wxt enhanced
gnuplot> set xlabel "X"
gnuplot> set ylabel "Y"
gnuplot> set xrange [0:10]
gnuplot> set xtics 0,1,10
gnuplot> unset key
gnuplot> set title "0阶贝塞尔函数 J_0(x)"
gnuplot> plot besj0(x)
```



这里的 besj0(x) 就是 gnuplot 里面预定义的 0 阶贝塞尔函数。如果现在请您从这个图上估计出 [0, 10] 内 $J_0(x)$ 的零点数值，也就是方程 $J_0(x) = 0$ 的解，恐怕您很难说的准确。但是如果为这个图加上栅格（grid），就容易多了：

```
gnuplot> set grid
gnuplot> replot
```

这时我们很容易估计出三个零点的数值：2.4, 5.5, 8.6。通过查表我们可以知道，这三个零点比较精确的数值分别为 2.4048, 5.5201, 8.6537。这和我们的估计值差不太多。如果我们想更精确的估计数值，可以尝试改一下 xrange：

```
gnuplot> set xrange [8:9]
gnuplot> set xtics 8,0.1,9
gnuplot> replot
```

这相当于把图像在零点附近放大了。把鼠标放在画图区域，画图框左下角就会显示出鼠标所在位置的坐标。现在我们把鼠标放在函数图线和 X 轴的交叉点上，左下角显示的横坐标为 8.65243，这和我们查表所得的数值更接近了。

如果想进一步让结果精确一些，我们可以利用 gnuplot 的计算功能。我们可以通过尝试计算的方法获得方程的数值解：

```
gnuplot> print besj0(8.65)
0.00101216621937318
gnuplot> print besj0(8.66)
-0.0017019446057587
gnuplot> print besj0(8.6537)
7.5770361108123e-06
gnuplot> print besj0(8.6536)
3.47225104115535e-05
gnuplot> print besj0(8.6538)
-1.95681245811775e-05
```

所以在 8.6 附近，$J_0(x) = 0$ 精确到小数点后 4 位的数值解为 8.6537，这和我们查表的结果一模一样。由于我们已经通过图像知道了数值解的大概位置，再加上合理利用线性插值，我们可以很快得到精确的结果。

14 第二坐标轴

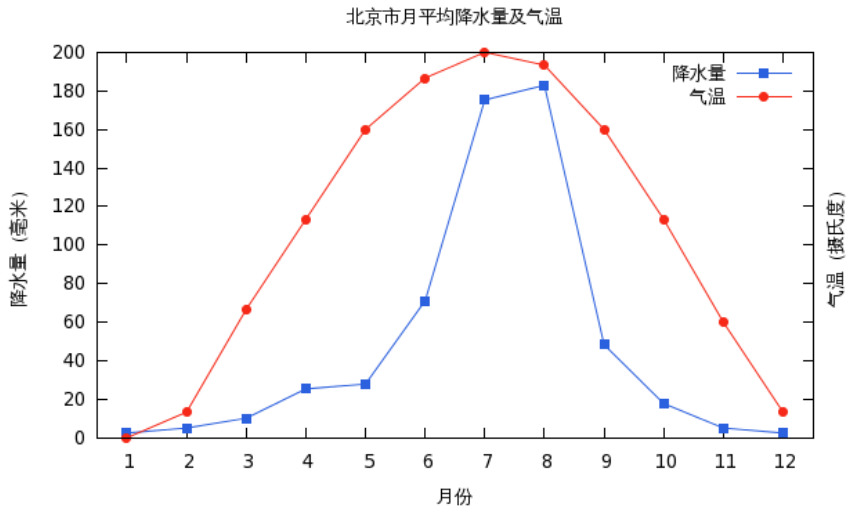
回首看看我们以前所有的作图，横坐标都标示在底部，而纵坐标都标示在左侧。其实，在图像顶部和右侧，还隐藏着一对不太引人注意的坐标轴，我们可以管它们叫做“第二坐标轴”。平时，它们只是第一对坐标轴的镜像；在我们需要的时候，它们可以用来表示不同的物理量。有时候，我们会有两组性质不同但是又相互关联的数据，这时候我们或许想把他们画在同一副图上，以便比较。

还拿北京市月平均降水量举例，但是这次，我们把温度也加上。下面是我们的数据文件 weather_beijing.dat：

```
### 文件开始 ###
# 北京月平均降水量（毫米）及气温（摄氏度）
#
# 月份   降水量   气温
# ===================
1       2.5     -4
2       5.1     -2
3       10.2    6
4       25.4    13
5       27.9    20
6       71.1    24
7       175.3   26
8       182.9   25
9       48.3    20
10      17.8    13
11      5.1     5
12      2.5     -2
### 文件结束 ###
```

我们之前讲过的所有有关坐标的参数，在第二坐标轴上均适用，只不过相应的名字起始字母改为 x2 或者 y2，例如 ylabel 改为 y2label。另外，plot 命令有一个新的参数 axis，用来控制使用哪个坐标轴，例如 axis x1y2 就表示使用第一横轴和第二纵轴。现在我们来看用上面数据作图的例子：

```
gnuplot> set xlabel "月份"
gnuplot> set ylabel "降水量（毫米）"
gnuplot> set y2label "气温（摄氏度）"
gnuplot> set title "北京市月平均降水量及气温"
gnuplot> set xrange [0.5:12.5]
gnuplot> set xtics 1,1,12
gnuplot> plot "weather_beijing.dat" u 1:2 w lp pt 5 lc rgbcolor "#2B60DE"\
> axis x1y1 t "降水量", "weather_beijing.dat" u 1:3 w lp pt 7\
> lc rgbcolor "#F62817" axis x1y2 t "气温"
```

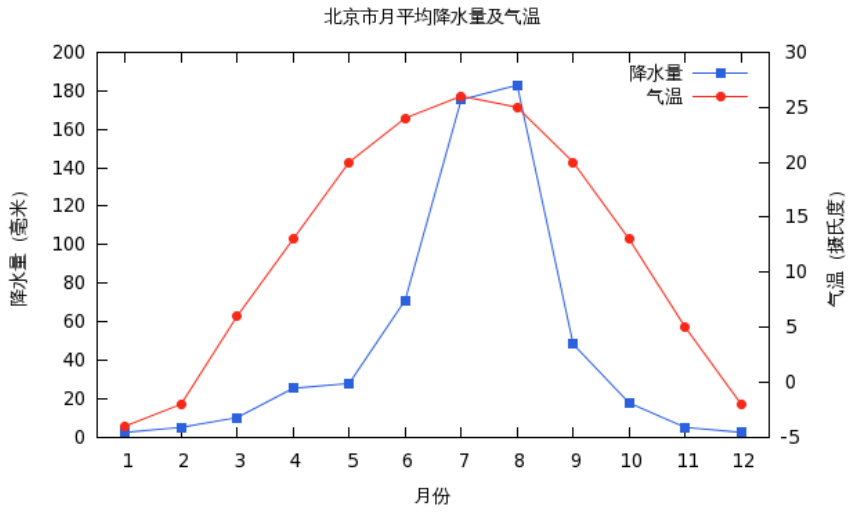


这里的气温数据使用了图像右边的第二纵轴 y2，但是 y2 轴上的刻度并没有变化，依然是左边 y1 轴的镜像。我们在这里有两件事要做：

1. 去除右边纵轴上的 y1 刻度镜像，否则这些刻度标记将和新的 y2 刻度标记混起来，导致无法识别；
2. 在右边纵轴上加上 y2 刻度标记。

我们执行下面的命令：

```
gnuplot> set ytics nomirror
gnuplot> set y2tics
gnuplot> replot
```



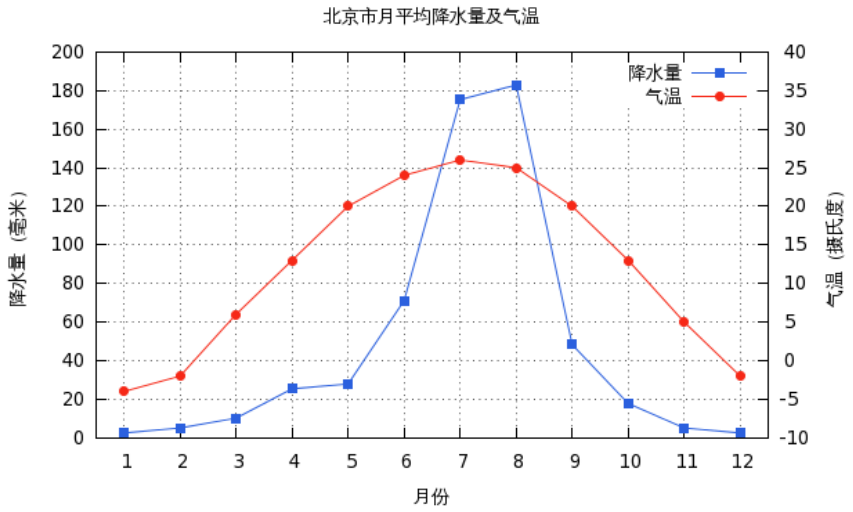
好了，现在降水量和温度数据分别对应于左侧和右侧的纵坐标。

看到这里，我们可能有点怀念我们上一讲谈到的 grid。如果能加上栅格，数据图示就更清楚了。但是现在我们有两组不同的纵坐标，如果都开启栅格，还不乱套了？set grid 命令允许我们在开启栅格时，选择使用哪一组坐标。例如：

```
gnuplot> set grid xtics y2tics
```

会开启 x1 和 y2 的栅格。但是这还是不能兼顾两组数据。最好的解决方案是，让两个纵轴有相同数目的分格，这样两套 grid 也就重合了，开启任何一个就可以了。例如，上面的图中左侧纵轴有 10 个分格，我们让右侧纵轴也有 10 个分格：

```
gnuplot> set y2range [-10:40]
gnuplot> set y2tics 5
gnuplot> set grid
gnuplot> replot
```



现在看起来好多了。

最后，不知道大家注意到没有，在开始的 plot 命令里，我们用了新的方式定义图线颜色。在第七讲“点线风格”里，我们提到过，可以用预定义的数字代码来定义图线颜色。但是在这里，我们使用了 rgbcolor 来定义颜色，这很大程度上增加了颜色选择范围，允许更好的显示效果。而其用法也很简单，就是在 rgbcolor 之后，加上颜色的 RGB 代码，了解 HTML 的朋友应该对这个不陌生。

15 Gnuplot 的坐标系统及标签

我们现在知道了 gnuplot 有第一（first）和第二（second）两套坐标系统，但是 gnuplot 的坐标系统还不止于此。除此之外，它还有 graph，screen 和 character 三套坐标系统。

graph 和 screen 都是归一化的坐标系统。graph 以坐标轴包围区域为界，左下角为 0,0，右上角为 1,1；screen 以整个图片区域为界，左下角为 0,0，右上角为 1,1。

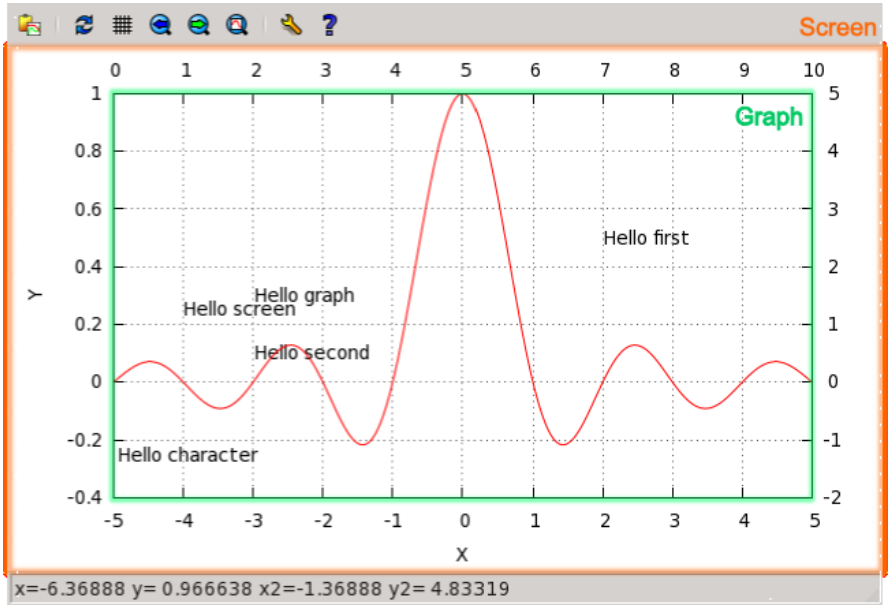
character 顾名思义，是以字符大小为单位长度的坐标系统，因此它的单位长度依赖于字体大小。它的原点位置和 screen 相同。

下面我们结合 label 命令来了解一下这几个坐标系统。我们之前讲过 xlabel 和 ylabel。而这里的 label 命令，是在图中任何地方插入文字标签。还是来看例子：

```
gnuplot> sinc(x) = sin(pi*x)/(pi*x)
gnuplot> set xlabel "X"
gnuplot> set ylabel "Y"
gnuplot> unset key
gnuplot> set samples 500
```

```
gnuplot> set xrange [-5:5]
gnuplot> set xtics 1
gnuplot> set x2range [0:10]
gnuplot> set x2tics 1
gnuplot> set y2range [-2:5]
gnuplot> set y2tics 1
gnuplot> set grid
gnuplot> set label 1 "Hello first" at 2,0.5
gnuplot> set label 2 "Hello second" at second 2,0.5
gnuplot> set label 3 "Hello graph" at graph 0.2,0.5
gnuplot> set label 4 "Hello screen" at screen 0.2,0.5
gnuplot> set label 5 "Hello character" at character 10,5
gnuplot> plot sinc(x)
```

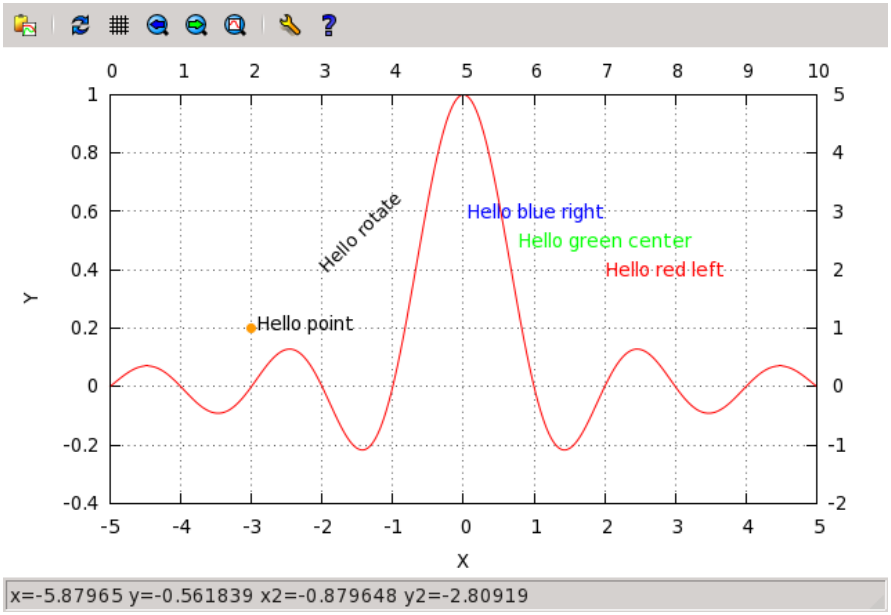
这里我们画一个 sinc 函数图像。为了说明问题，我们把第二坐标系也都标示了出来，虽然函数图像并没有用到第二坐标。其他命令前面都讲过了，这里只看五个 set label 命令。set label 之后紧跟的那个整数，就是一个标识符，用以区别各个 label，可以随便选个整数。在字符串之后，at 参数指定标签坐标。默认为 first 坐标系统，也可以使用其它坐标系统。下面是生成的图片：



为了帮助大家理解，我们把 graph 和 screen 各自的坐标区域分别用绿色和橙色表示了出来。

标签文字的默认对齐方式为居左，也就是指定的坐标位置在文字的左边。我们也可以在 label 命令里选择其他对齐方式。除此之外，我们还可以在 label 命令里指定文字颜色，旋转文字，或者在指定坐标位置处加一个点。下面例子中的每个参数不必一一解释了，因为和我们前面接触过的命令都是一致的：

```
gnuplot> set label 1 "Hello red left" at 2,0.4 left textcolor rgb "#FF0000"
gnuplot> set label 2 "Hello green center" at 2,0.5 center textcolor rgb "#00
   FF00"
gnuplot> set label 3 "Hello blue right" at 2,0.6 right textcolor rgb "#0000FF"
gnuplot> set label 4 "Hello rotate" at -2,0.4 rotate by 45
gnuplot> set label 5 "Hello point" at -3,0.2 point pt 7 lc rgb "#FF9900"
gnuplot> replot
```



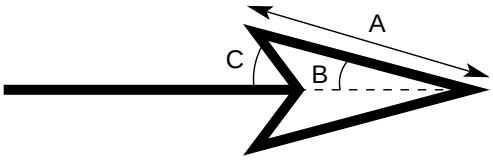
16 箭头

有了坐标系的知识打底，其他很多东西很好谈了。我们的图上除了标签之外，还有一个常用的标志：箭头。关于箭头的命令是 set arrow，语法和 label 有些类似，包括以下这些常用参数：

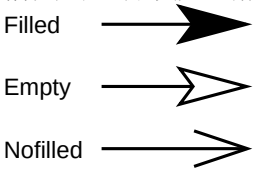
- `from ... to ...`
  箭头的起点和终点坐标。如果把 to 换成 rto，第二个坐标就表示相对位置而不是绝对坐标。

- `nohead, head, backhead, heads`
  分别表示：没有箭头（其实就是线段），箭头在终点，箭头在起点，双向都有箭头。

- `size <length>,<angle>,<backangle>`
  箭头尺寸，默认长度单位为 first 坐标单位长度。
  下图中 A，B，C 分别代表 <length>，<angle>，<backangle>。



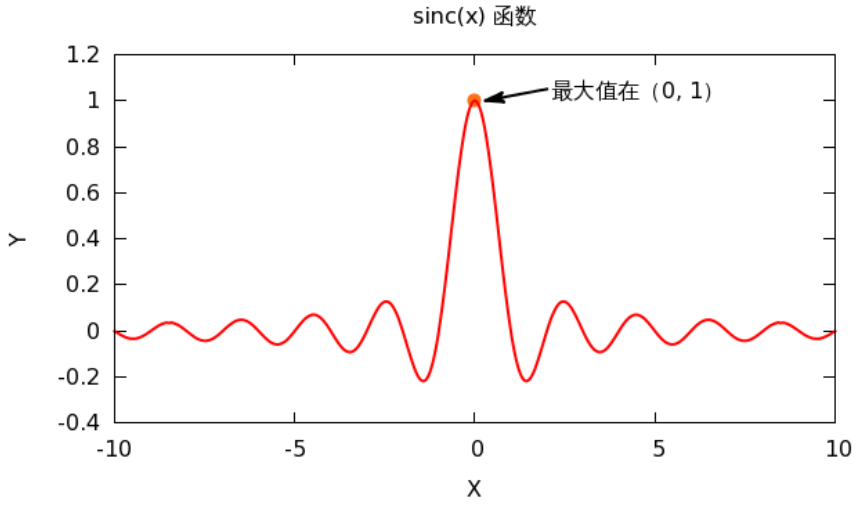
- `filled, empty, nofilled`
  箭头的三种填充风格：



下面我们看例子，还是画 sinc(x) 函数：

```
gnuplot> set term wxt font "DejaVu␣Sans,12"
gnuplot> sinc(x) = sin(pi*x)/(pi*x)
gnuplot> set xlabel "X"
```

```
gnuplot> set ylabel "Y"
gnuplot> set yrange [-0.4:1.2]
gnuplot> set title "sinc(x)␣函数"
gnuplot> unset key
gnuplot> set samples 500
gnuplot> set arrow 1 from 2,1.05 to 0.3,1 filled size 0.5,15,60 lw 2
gnuplot> set label 1 at 0,1 point pt 7 ps 1.5 lc rgb "#F87217"
gnuplot> set label 2 "最大值在 (0, 1) " at 2.1,1.05
gnuplot> plot sinc(x) lw 2
```



17 边框和坐标轴

我们现在所有绘图的坐标刻度均标在图像边框上，无论上下左右。这样做的好处是函数或数据图线清楚，不会和坐标标注混在一起。其实，我们小时候数学课上最早学习坐标系的时候，都是让 X 轴和 Y 轴正交于原点，而刻度标注在坐标轴上。这样的图像在定性表现函数关系，尤其有一定对称性的函数关系时，比较一目了然。

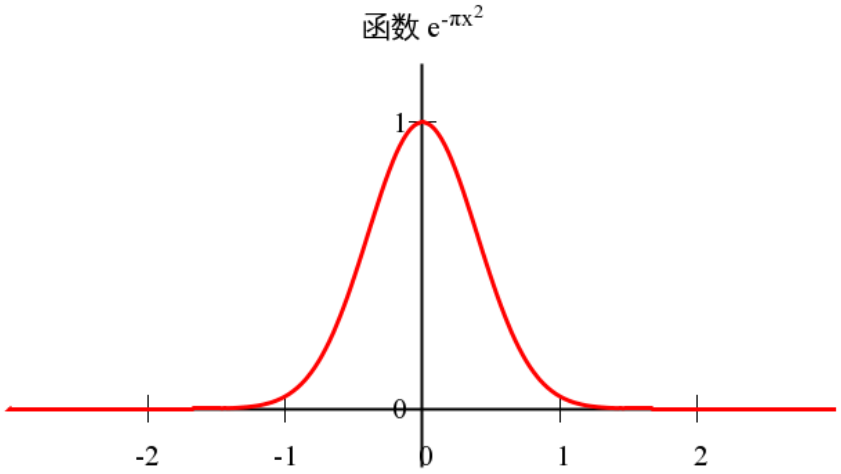
让我们来看看怎样用 gnuplot 得到这样的效果。

1. 用 unset border 命令把边框去掉;
2. 用 set zeroaxis 命令画出正交于原点的坐标轴;
3. 在设定坐标刻度时加上 axis 参数，这样刻度会出现在坐标轴上面，而不是边框上。

为了避免审美疲劳，这次我们拿高斯函数举个例子:

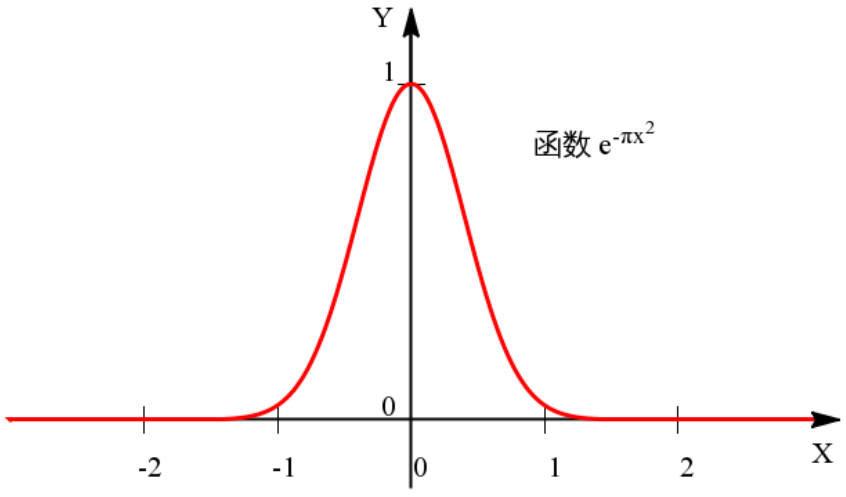
```
gnuplot> set term wxt enhanced font "Times␣New␣Roman,16"
gnuplot> gauss(x) = exp(-pi*x*x)
gnuplot> set title "函数 e^{-πx^2}"
gnuplot> set samples 500
gnuplot> set xrange [-3:3]
gnuplot> set yrange [-0.2:1.2]
gnuplot> unset key
gnuplot> unset border
```

```
gnuplot> set zeroaxis lt -1 lw 2
gnuplot> set xtics axis -2,1,2
gnuplot> set ytics axis 0,1,1
gnuplot> plot gauss(x) lw 3
```



例子中的参数前面都介绍过，如果不记得了，可以复习一下“坐标取值范围及刻度”和“点线风格”等章节。这里的图像已经很像模像样了，除了标签位置还不那么理想，而且坐标轴没有箭头。幸好，我们上一讲刚刚谈到过箭头，下面来试试做一下微调：

```
gnuplot> set title "函数 e^{-πx^2}" offset 12,-5
gnuplot> set xtics axis -2,1,2 offset 0.4,0
gnuplot> set ytics axis 0,1,1 offset 0,0.4
gnuplot> set arrow 1 from 2,0 to 3.2,0 filled size 0.2,15,60 lw 2
gnuplot> set arrow 2 from 0,1 to 0,1.22 filled size 0.2,15,60 lw 2
gnuplot> set rmargin 4
gnuplot> set label 1 "X" at 3.0,-0.1
gnuplot> set label 2 "Y" at -0.3,1.2
gnuplot> replot
```



这里有几个命令同时用到了新的参数：offset。它的作用就是把命令里提到的标签文字平移一段距离。在这里，offset 默认的坐标系统是 character。我们慢慢会体会到这种做法的好处，它使得我们很多时候改变字体大小，而不必重新设置 offset。

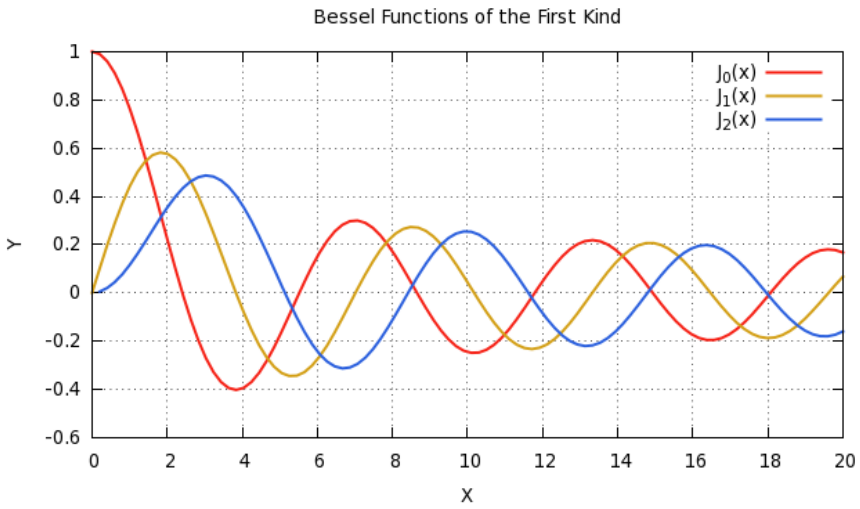
另外，set rmargin 命令用于设置图像右边空白宽度，单位也是 character。一般情况下，四边空白宽度都是自动设置的。现在我们在右边增加了箭头，而绘图显示区域不会因此自动扩大，这样会导致箭头无法完整显示，所以要手动改一下设置。相应的，上、左、下边的空白宽度，分别由 tmargin，lmargin，bmargin 参数控制。

18 图例

在同一图像中包含多组数据或函数时，图例是必要的。我们这一次谈一谈图例的微调。

这次来画前 3 阶的第一类贝塞尔函数 $J_n(x)$。在 gnuplot 里，0 阶和 1 阶贝塞尔函数已经定义了，分别为 besj0(x) 和 besj1(x)，而 2 阶贝塞尔函数可以通过递推关系构造出来。下面是例子：

```
gnuplot> set term wxt enhanced
gnuplot> besj2(x) = besj1(x)*2/x - besj0(x)
gnuplot> set xrange [0:20]
gnuplot> set xtics 2
gnuplot> set xlabel "X"
gnuplot> set ylabel "Y"
gnuplot> set title "Bessel Functions of the First Kind"
gnuplot> set grid
gnuplot> set style line 1 lw 2 lc rgb "#F62217"
gnuplot> set style line 2 lw 2 lc rgb "#D4A017"
gnuplot> set style line 3 lw 2 lc rgb "#2B60DE"
gnuplot> plot besj0(x) ls 1 t "J_0(x)", besj1(x) ls 2 t "J_1(x)", besj2(x) ls 3
    t "J_2(x)"
```

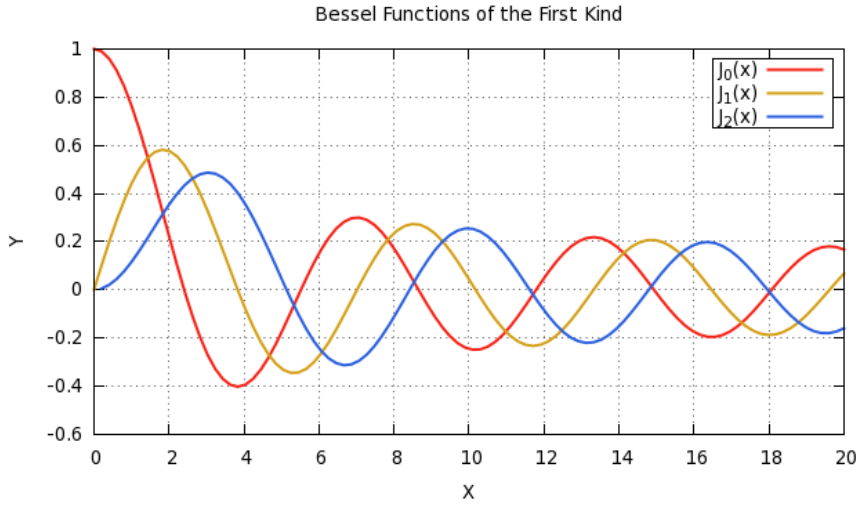


之前我们讲过，plot 命令后面可以跟随一些参数（例如 linewidth, linecolor 等）来改变点线风格。在上面的例子中，我们把这些参数单独拿出来放到了 set style 命令里，定义了三个 linestyle，然后在 plot 命令里再调用这些 linestyle。这样子做和我们之前的做法效果上没什么不同，唯一的区别是让 plot 命令短了一些。另外，改变风格可能容易一点。

上面是默认的图例，下面让我们进行微调。

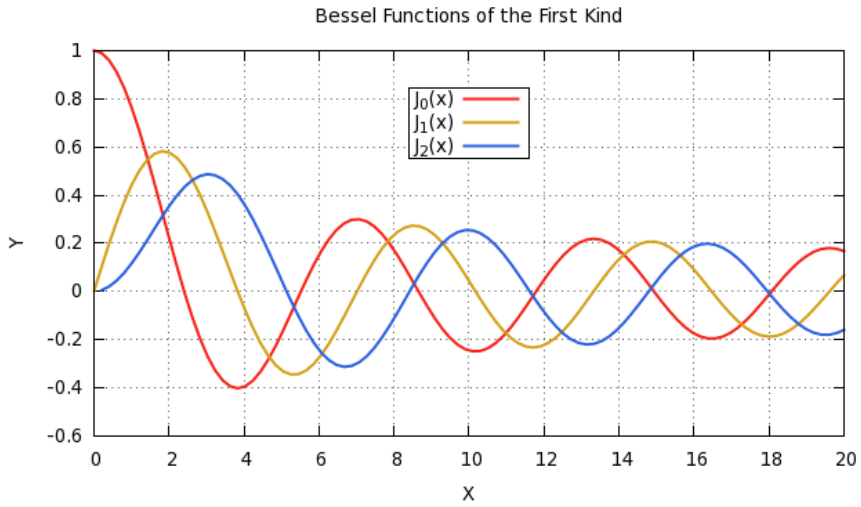
1. 为图例加上边框

```
gnuplot> set key box
gnuplot> replot
```



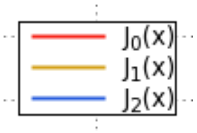
2. 改变图例显示位置

```
gnuplot> set key center at 10,0.7
gnuplot> replot
```



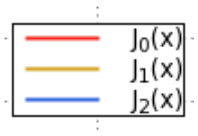
3. 把图例的 title 和图线示例调换位置

```
gnuplot> set key reverse
gnuplot> replot
```



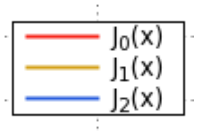
4. 调整图例边框宽度 width（或高度 height）

```
gnuplot> set key width 1
gnuplot> replot
```



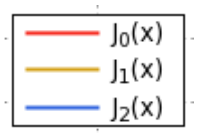
5. 调整 title 文字对齐方式（Left 或者 Right，注意首字母大写）

```
gnuplot> set key Left
gnuplot> replot
```



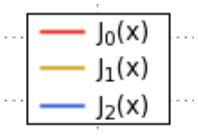
6. 调整图例行间隔

```
gnuplot> set key spacing 1.2
gnuplot> replot
```



7. 调整图线示例长度

```
gnuplot> set key samplen 2
gnuplot> replot
```

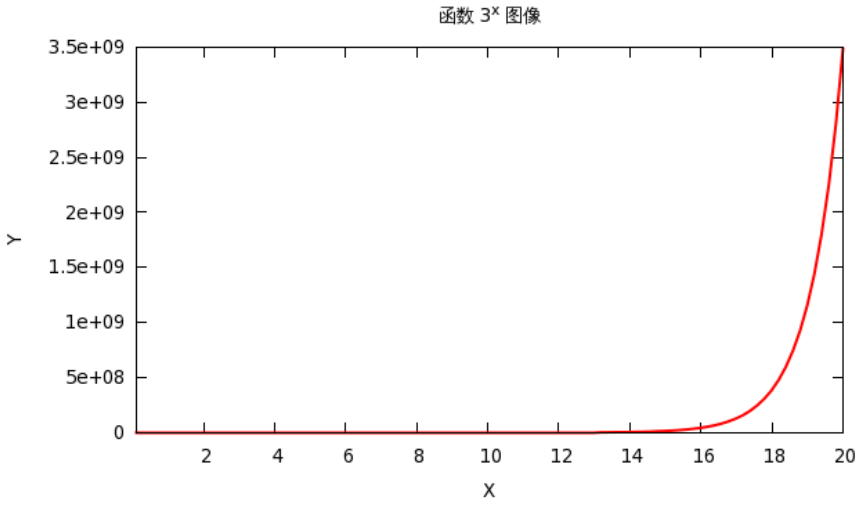


这些并不是 set key 的全部参数。在 gnuplot 里，如果想深入了解任何命令的详细用法，不要忘记使用 help 命令。

19 对数坐标

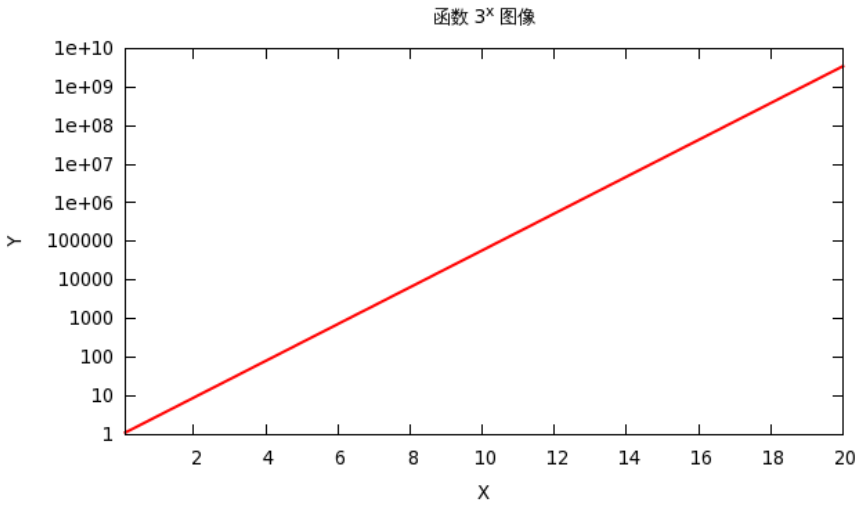
我们这次来谈谈怎样在 gnuplot 里面绘制对数坐标图。我们先在一般直角坐标系里画一下函数 3x 的图像:

```
gnuplot> set term wxt enhanced
gnuplot> set xlabel "X"
gnuplot> set ylabel "Y"
gnuplot> unset key
gnuplot> set title "函数 3^x 图像"
gnuplot> set xrange [0.1:20]
gnuplot> plot 3**x lw 2
```



下面把 y 轴改为对数坐标：

```
gnuplot> set logscale y
gnuplot> replot
```

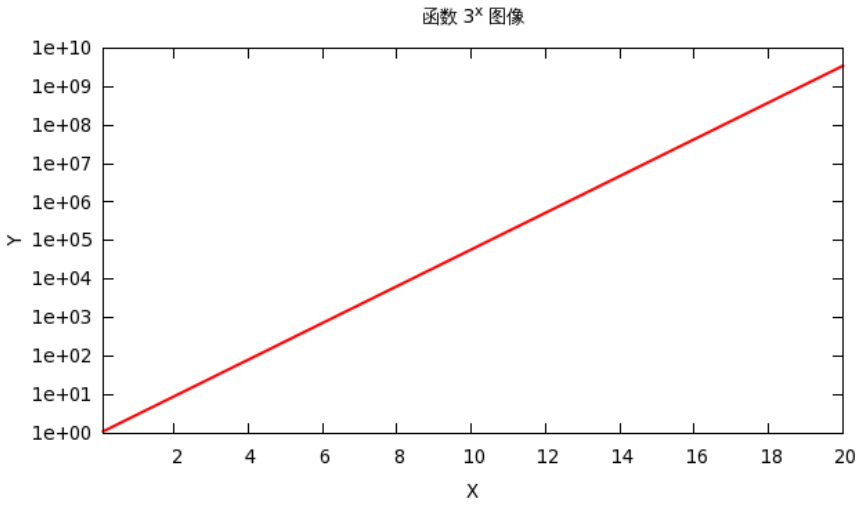


set logscale 命令用于指定对数坐标轴，这个例子中是 y 轴。如果要使用双对数坐标，只需执行：

```
set logscale xy
```

这里 y 轴的数字标注有点不好看，因为默认情况下 gnuplot 根据数字大小自动选择数字格式。我们希望用一致的方式，例如都以指数方式显示数字。我们可以用 set format 命令来指定数字显示格式，使用方法为 set format 加上坐标轴，再加上表示格式的字符串：

```
gnuplot> set format y "%.0e"
gnuplot> replot
```



后面表示数字格式的字符串，对于熟悉 C 语言的读者应该不陌生，它和 printf 函数中表示数字格式的字符串非常类似。常见的有以下几个：

- `%f` 小数格式
- `%e` 指数格式
- `%g` 根据长度自动选择 `%f` 或者 `%e`
- `%t` 指数格式的有效数字部分
- `%T` 指数格式的指数部分

另外，各特殊字符之前的数字可以用于表示有效数字的精度。

20 图像尺寸

当我们在 gnuplot 里面执行：

```
gnuplot> set term pngcairo
```

我们会发现有如下的返回信息：

```
Terminal type set to 'pngcairo'
Options are '␣size␣640,␣480␣'
```

这表示输出的 png 图片默认大小是 640x480 像素。如果我们想改变输出图片的大小，可以在 set term 命令之后加上 size 参数，例如：

```
gnuplot> set term pngcairo size 800,600
```

对于 eps 和 pdf 输出，默认的 size 单位是英寸，而不是像素。这是因为 eps 和 pdf 均是矢量图片，像素值没什么意义。

除了在设置 terminal 的时候可以指定 size 参数，gnuplot 里面还有一个单独的 set size 命令。这两种设置方法的区别在于：

- set terminal 的时候 size 参数指定的是整个图片的尺寸，包括标签、标题、四边空白等等，而 set size 命令指定的仅仅是绘图区域的尺寸；

- set terminal 的时候 size 参数指定的是绝对尺寸，例如像素、英寸等，而 set size 命令指定的是相对尺寸，也就是绘图区域相对于整个图片大小的比例。例如 set size 0.5,0.5 时，绘图区域仅占整个图片大小的四分之一。

由于上述第二点，set size 命令更常用的形式是 set size ratio，这时只需给出绘图区域高和宽的比例，例如：

```
gnuplot> set size ratio 0.5
```

将会画出高宽比为 1:2 的图像。而高宽比为 1 的图像（也就是正方形），还有一个单独的设置方法：

```
gnuplot> set size square
```

在 gnuplot 旧的版本中，这两种设置尺寸的方法比较混乱。从 gnuplot 4.4 起，基本上都统一为上述方式。

21 极坐标

gnuplot 可以在极坐标下绘图，相应命令是：

```
gnuplot> set polar
```

然后会出现下面的返回信息：

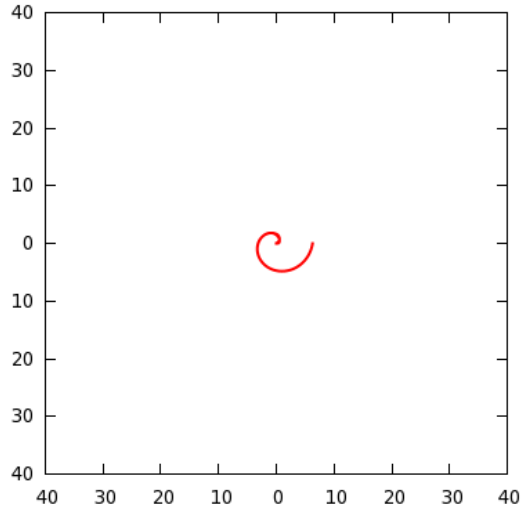
```
        dummy variable is t for curves
```

这表明，在极坐标下，自变量名字是 t，这不同于直角坐标下的 x，需要注意。t 本身代表角度，默认单位是弧度（radians）。如果您想使用角度（degrees），可以执行下面的命令：

```
gnuplot> set angles degrees
```

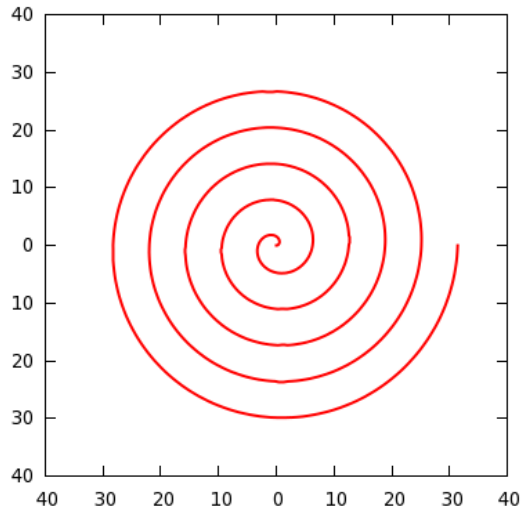
下面我们看例子：

```
gnuplot> set polar
gnuplot> unset key
gnuplot> set samples 1000
gnuplot> set xrange [-40:40]
gnuplot> set yrange [-40:40]
gnuplot> set size square
gnuplot> plot t lw 2
```



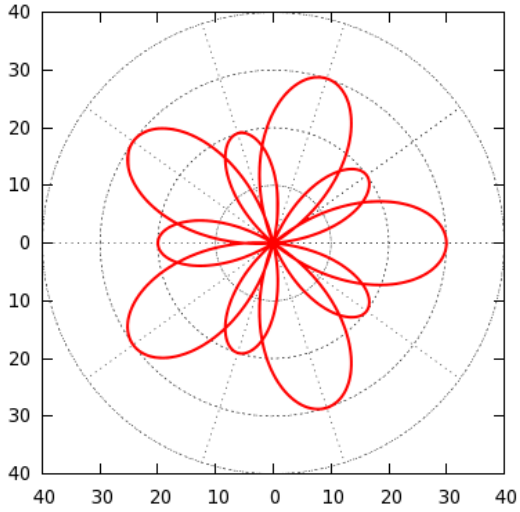
从这里我们看出，t 默认的取值范围是是 [0:2*pi]。和直角坐标类似，改变 t 取值范围的命令是 set trange:

```
gnuplot> set trange [0:10*pi]
gnuplot> replot
```



极坐标下的栅格和直角坐标不同，应该是按一定角度分隔的扇区，所以 set grid 命令需要加上 polar 参数。默认扇区分隔角度是 30 度，该角度可以作为 set grid polar 的参数进行调整:

```
gnuplot> set grid polar pi/5
gnuplot> plot 5+25*cos(5*t/2) lw 2
```



22 参数方程

gnuplot 也能画参数方程。首先设置参数方程环境:

```
gnuplot> set parametric
```

然后我们会看见返回信息:

```
        dummy variable is t for curves, u/v for surfaces
```

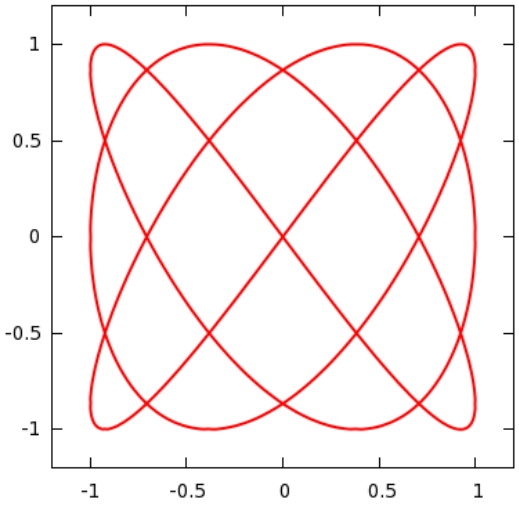
和极坐标类似，参数方程的自变量也是 t。后面的 u/v 是用于 3D 绘图的参数方程自变量，我们目前暂时不管它。

对于参数方程 x = f(t), y = g(t)，绘图命令为

```
plot f(t),g(t)
```

下面我们看一个例子，这是一个李萨如（Lissajous）曲线:

```
gnuplot> set parametric
gnuplot> set xrange [-1.2:1.2]
gnuplot> set yrange [-1.2:1.2]
gnuplot> set trange [0:2*pi]
gnuplot> set samples 1000
gnuplot> set size square
gnuplot> unset key
gnuplot> plot sin(3*t),sin(4*t) lw 2
```



23 误差条

Error bar 是在图像上表现数据误差范围的一种方式。对于含有误差项的数据，除了通常的 x 轴和 y 轴两列数据外，我们还需要有额外的误差数据列。

拿 x 数据列举例，如果误差用标准差 σ_x 来表示，那么数据取值范围可以表示为 $[x-\sigma_x, x+\sigma_x]$，这时候只要增加一列误差项就行了，所以一共需要 3 列数据。如果误差用最小值 xmin 和最大值 xmax 来表示，那么数据取值范围可以表示为 $[x_{min}, x_{max}]$，这时候需要增加两列误差项，所以一共需要 4 列数据。对于 y 数据误差，表达方法和 x 类似。如果同时包含 x 和 y 误差，就需要把两者结合起来。

在 gnuplot 里，error bar 的基本使用方法是：

```
plot "数据文件名" using <using参数> with <xerrorbars | yerrorbars | xyerrorbars>
```

using 命令在之前的“多组数据绘图”博文里已经介绍过，目的是选择哪些列数据进行绘图，数据列数必须和后面选择的绘图方式对应。with 命令后面跟的是绘图方式，选择用 xerrorbars，yerrorbars，还是 xyerrorbars。根据不同绘图方式，所需数据列数分别为：

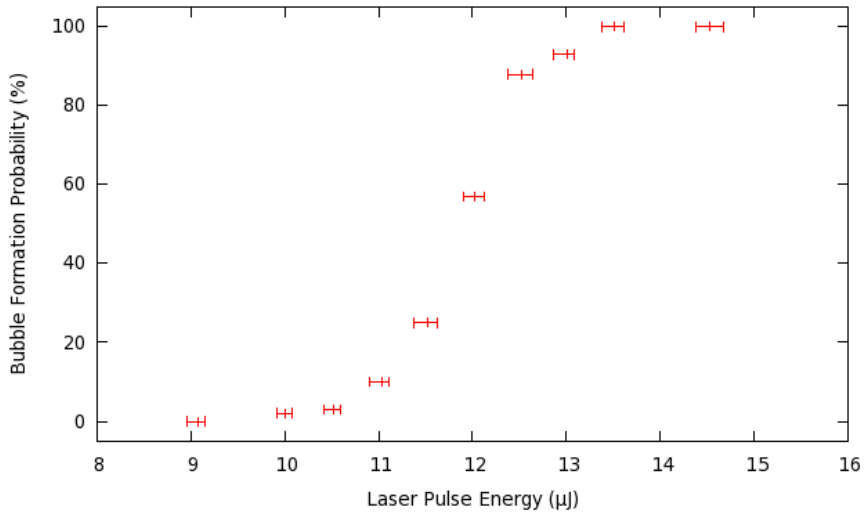
- xerrorbars
  - 3 列：x y σ_x
  - 4 列：x y x_{min} x_{max}
- yerrorbars
  - 3 列：x y σ_y
  - 4 列：x y y_{min} y_{max}
- xyerrorbars
  - 4 列：x y σ_x σ_y
  - 6 列：x y x_{min} x_{max} y_{min} y_{max}

下面我们举一个例子，这是一个测量在液体中聚焦的脉冲激光在焦点处产生气泡几率的实验，数据文件（probability.dat）如下：

```
### 文件开始 ###
# Ave Energy     Probability     Min Energy      Max Energy      Energy SD
# (micro J)      (%)             (micro J)       (micro J)       (micro J)
# ============================================================================
9.08             0               8.96            9.15            0.06
10.00            2               9.91            10.08           0.05
10.52            3               10.41           10.60           0.06
11.03            10              10.90           11.11           0.06
11.52            25              11.38           11.62           0.07
12.03            57              11.90           12.13           0.07
12.52            88              12.38           12.64           0.08
13.01            93              12.86           13.09           0.07
13.51            100             13.38           13.61           0.08
14.52            100             14.38           14.67           0.08
### 文件结束 ###
```

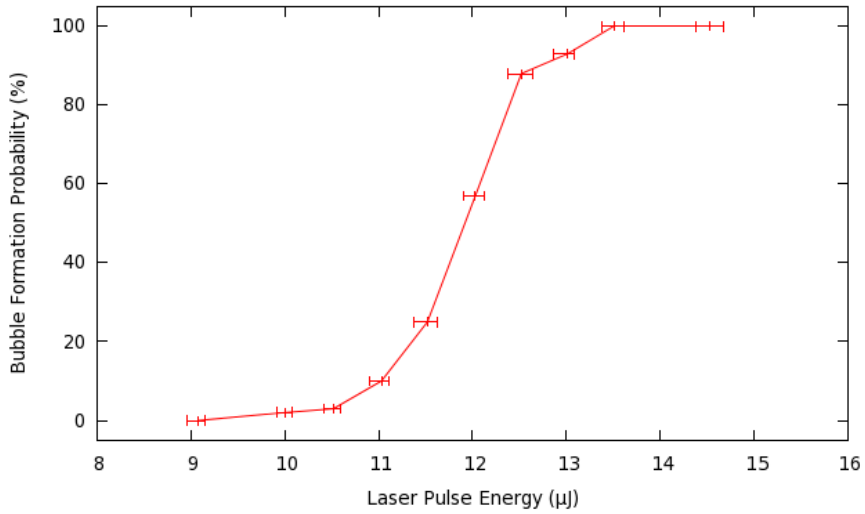
x 轴数据为激光能量，y 轴数据为气泡产生几率，这里只有 x 误差，并且同时包含了最小最大值和标准差。我们现在用最小最大值画图：

```
gnuplot> set xrange [8:16]
gnuplot> set yrange [-5:105]
gnuplot> unset key
gnuplot> set xlabel "Laser Pulse Energy (μJ)"
gnuplot> set ylabel "Bubble Formation Probability (%)"
gnuplot> plot "probability.dat" using 1:2:3:4 with xerrorbars
```



如果既要画 error bar，又要连线，可以把上述命令中的 errorbars 换为 errorlines：

```
gnuplot> plot "probability.dat" using 1:2:3:4 with xerrorlines
```



24 拟合

gnuplot 除了绘图功能之外，最简单实用的功能就是拟合了。gnuplot 可以进行单变量甚至多变量的线性和非线性拟合。虽然可能不像专门的数学软件那么强大，但是足以对付日常需要了。我们拿上一讲里的数据来举例子。

首先，要定义一个待拟合的函数：

```
gnuplot> f(x)=50*(1+erf(a*(x-b)))
```

这里使用了误差函数 erf(x)，有两个待定的参数：a, b。下面我们生成一个文件“fit.par”，里面包含的是参数 a 和 b 的初值：

```
a = 1
b = 12
```

初值的选择要尽可能贴近结果，否则可能导致误差甚至无法收敛。下面我们进行拟合：

```
gnuplot> fit [8:16] f(x) 'probability.dat' using 1:2 via 'fit.par'
```

gnuplot 里面关于拟合的命令是 fit，后面的自变量取值范围不是必需的。f(x) 函数已经在上面定义过了，数据文件“probability.dat”也已经在上一篇博文中交代过了。via 后面跟的是参数变量列表文件。执行 fit 命令之后，gnuplot 会输出一堆结果。我们忽略那些中间运算，只把最后结果贴在下面：

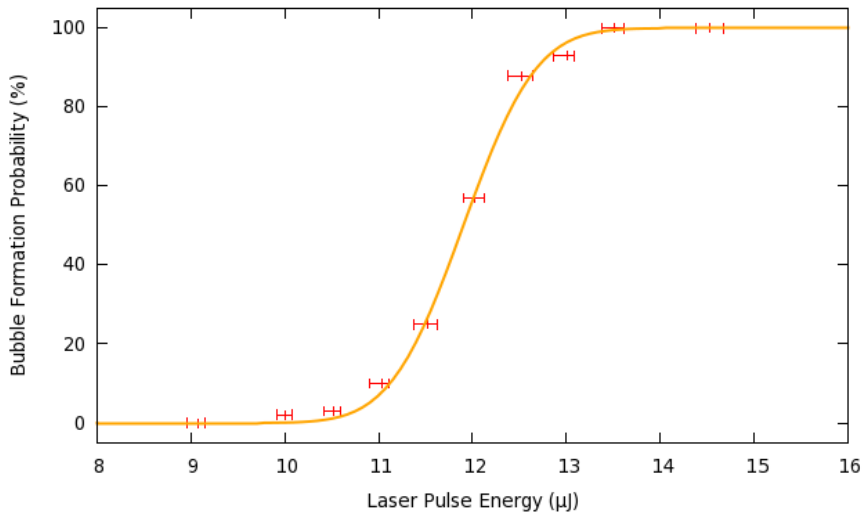
```
After 5 iterations the fit converged.
final sum of squares of residuals : 41.9399
rel. change during last iteration : -4.27973e-07

degrees of freedom    (FIT_NDF)                        : 8
rms of residuals      (FIT_STDFIT) = sqrt(WSSR/ndf)    : 2.28965
variance of residuals (reduced chisquare) = WSSR/ndf   : 5.24249

Final set of parameters            Asymptotic Standard Error
=======================            ==========================
```

```
a               = 1.15661          +/- 0.06331      (5.474%)
b               = 11.9027          +/- 0.02383      (0.2002%)


correlation matrix of the fit parameters:

                a      b
a               1.000
b               0.014  2.000
```

这段文字说明，经过 5 次迭代，gnuplot 得到了收敛的结果。中间部分是参数 a 和 b 的最终取值以及渐近标准差（asymptotic standard error）。渐近标准差的计算是基于线性拟合的，对于非线性拟合，渐近标准差一般都比真的标准差小，所以这个数字只能用于定性分析。而最后给出的相关矩阵（correlation matrix）可以帮助我们确认渐近标准差的可靠度，非对角元素绝对值越小，渐近标准差越接近真实标准差。

好了，现在我们可以把数据和拟合曲线画在同一张图上了：

```
gnuplot> set xrange [8:16]
gnuplot> set yrange [-5:105]
gnuplot> unset key
gnuplot> set xlabel "Laser Pulse Energy (μJ)"
gnuplot> set ylabel "Bubble Formation Probability (%)"
gnuplot> plot "probability.dat" using 1:2:3:4 with xerrorbars, f(x) lw 2 lc rgb
    "orange"
```

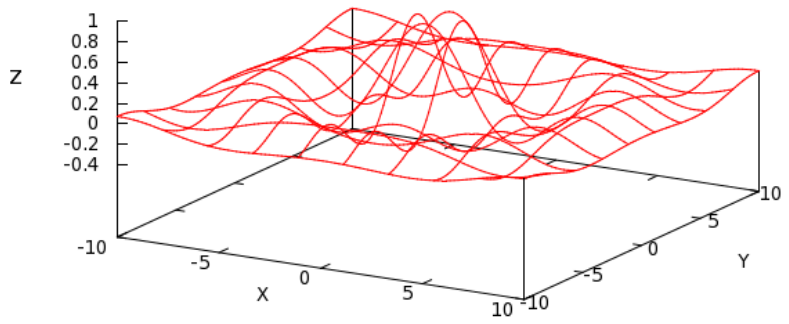


25 简单 3D 函数绘图

终于谈到 3D 绘图了。大多数情况下普通 3D 曲面绘图并不是一个好的选择，因为投影会使这样的绘图很难表达定量关系，通常只能定性的显示一下函数或者数据趋势。这种情况下，往往需要其他图像的配合才能展示定量的关系。我们从简单的函数图像入手，介绍一下 gnuplot 的 3D 绘图功能。

在 3D 情况下，gnuplot 的绘图命令是 splot：

```
gnuplot> f(x,y)=sin(sqrt(x*x+y*y))/sqrt(x*x+y*y)
gnuplot> set xlabel "X"
gnuplot> set ylabel "Y"
gnuplot> set zlabel "Z"
gnuplot> unset key
gnuplot> splot f(x,y)
```



显然，默认的网格有点过大，无法显示函数细节。这个网格大小可以通过 isosamples 参数来调节。另外，底部平面似乎离曲面太远了，留了很大一段空白，我们想把这个空白缩小一些。底部平面的位置由 xyplane 参数来控制，可以用：

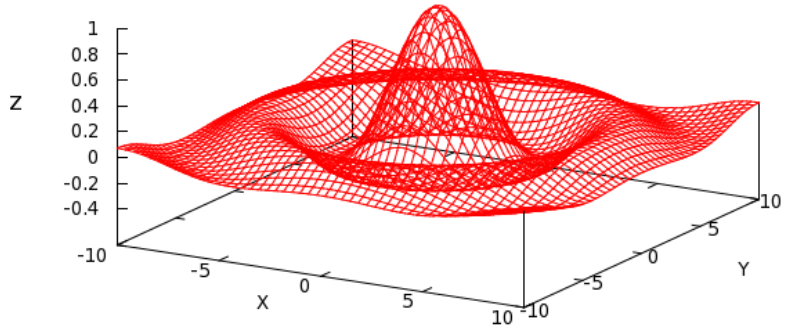
```
set xyplane at <z坐标>
```

来设定底部平面的绝对位置，也可以用：

```
set xyplane <相对比例>
```

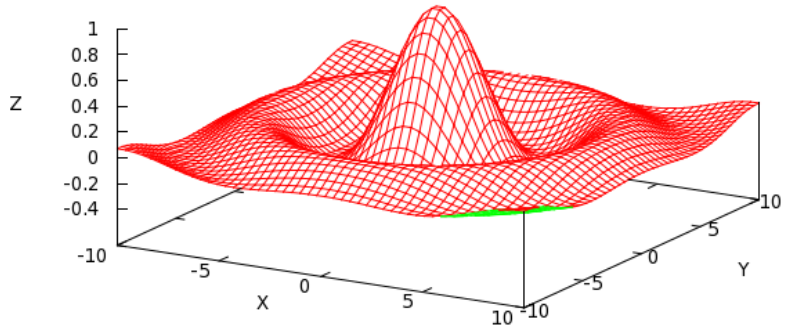
来设定空白部分相对于 zrange 的比例。 下面是例子：

```
gnuplot> set isosamples 50
gnuplot> set xyplane 0.2
gnuplot> replot
```



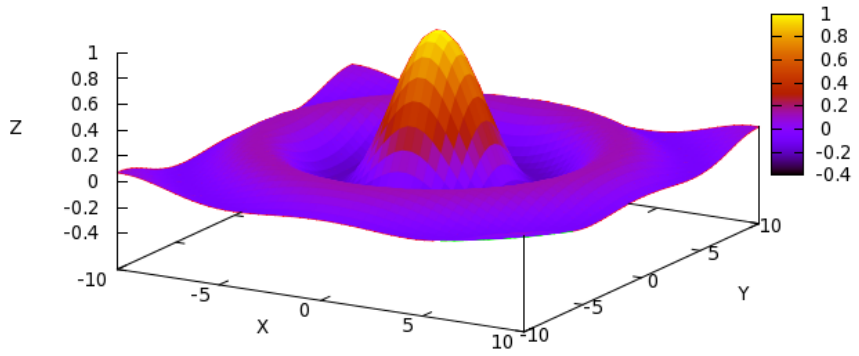
由于网格是透明的，因此网格重叠部分显得比较混乱。为了更清楚的显示函数图像，我们让后面被遮挡的部分隐藏起来：

```
gnuplot> set hidden3d
gnuplot> replot
```



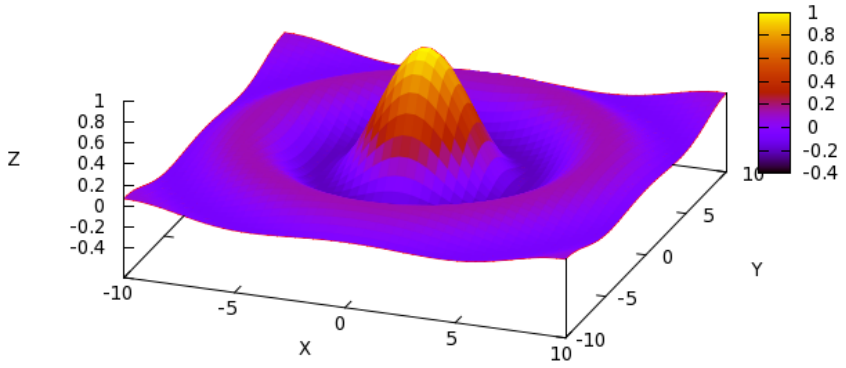
除了用网格表示曲面，我们还可以用色彩来表示不同的 z 值，gnuplot 把这种表示方式叫做 pm3d（palette-mapped 3d）：

```
gnuplot> set pm3d
gnuplot> replot
```



最后，既然是 3D 图像，就要涉及视角问题。gnuplot 里的 3D 视角用 set view 命令控制，默认视角是 60, 30，分别表示绕 x 轴和 z 轴的旋转角度。另外，一些 terminal 例如 wxt，支持直接用鼠标拖动图像改变视角。

```
gnuplot> set view 45,20
gnuplot> replot
```



26 数据文件存储格式

在 3D 数据绘图之前，我们先谈谈 gnuplot 数据文件的存储格式。为什么在 2D 绘图时我们没有涉及这个问题呢？因为 2D 绘图数据相对简单，只要有 x 和 y 两列数据就可以了。而 3D 绘图数据量成平方增长，数据存储方式也更多样化。

gnuplot 的数据可以以纯文本或者二进制方式存储。之前我们在 2D 绘图中均使用纯文本方式存储数据，这样的好处是简单直观，把数据按照 x 和 y 存为两列就可以了。在 3D 情况下，由于数据量激增，纯文本文件所占空间较大，读写速度也较慢，所以很多时候用二进制存储数据更方便。无论纯文本还是二进制，存储数据的方式都不是唯一的，我们这里只介绍最常用的方式。

（一）纯文本格式：

2D 的情况我们已经很熟悉了，这里说说 3D 的情况。3D 情况下，x, y, z(x,y) 可以按照下面格式存储：

```
<x0> <y0> <z0,0>
<x0> <y1> <z0,1>
<x0> <y2> <z0,2>
...  ...  ...
<x0> <yN> <z0,N>

<x1> <y0> <z0,0>
<x1> <y1> <z0,1>
...  ...  ...
```

这里有几个要点：

1. 每行存储 x, y, z 三个数据
2. 首先保持 x 不变，让 y 逐行变化
3. 当 y 遍历 [y0, yN] 取值范围后，x 取 [x0, xN] 里的下一个值，重复过程 2
4. 每个相同 x 取值的数据块之间，保留一个空行（这个比较容易忽视，需要注意）

然后就可以用 splot 命令绘制 3D 数据图了。

（二）二进制格式：

假设有 NxM 个数据（z值），数据逐行按照从左到右、从上到下的顺序存储在二进制文件中，这时候绘制数据文件的命令为：

```
plot '<filename>' binary <二进制参数列表>
splot '<filename>' binary <二进制参数列表>
```

可能有人会问，绘制 3D 数据不是用 splot 吗？为什么这里还会用 plot？我们暂且忽略这个问题，来看看二进制参数列表。

二进制文件只是一个数据流，gnuplot 可不知道我们数据的具体存储方式，这需要我们通过各个参数告诉 gnuplot。下面是一些常用参数：

1. array=(N,M)
   这是告诉 gnuplot 我们数据中 x 和 y 各自的维数

2. format
   我们的数据是 16 位还是 32 位？是整型还是浮点数？format 参数负责告诉 gnuplot 这些信息。我们可以通过执行下面的命令：

   ```
   show datafile binary datasizes
   ```

   来获得 format 参数的数据类型列表。

3. endian
   对于多字节数据，一般有两种不同的字节存储顺序：big 或者 little。如果您试图绘制一个二进制数据，而结果看起来莫名其妙，很可能就是 endian 设置错了。

4. dx, dy, dz
   这其实就是 Δx, Δy, Δz，也就是各个坐标的比例系数。例如设置 dx=2，而数据文件中 x 的取值范围是 [0:100]，那么所绘图像上 x 的取值范围就会变为 [0:200]。

下面是一个绘制二进制数据的例子：

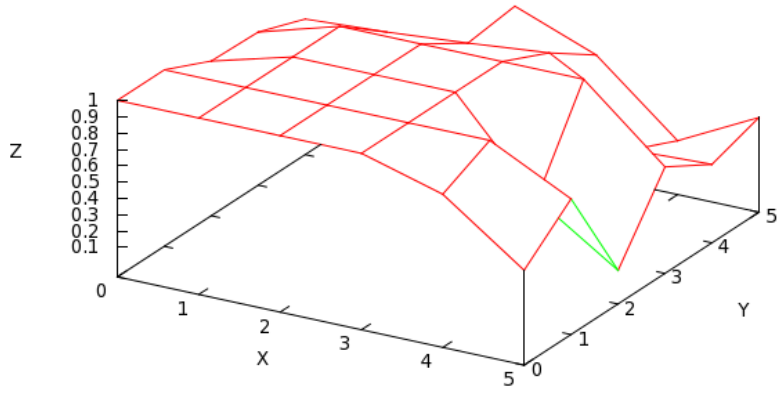
```
splot "datafile.bin" binary array=(64,64) endian=little format="%float" with
    lines
```

27 3D 数据曲面绘图及边框

理解了 gnuplot 的数据存储格式，我们可以来学习 3D 绘图了。我们拿一个 6X6 的数据举例，该数据以纯文本方式存在此文件中：data3d.dat

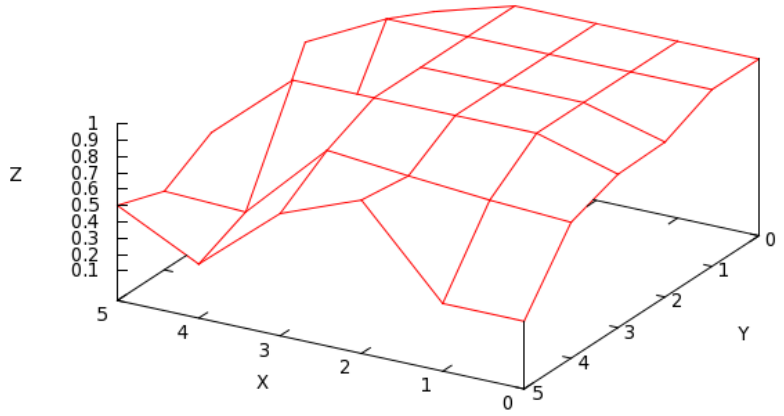
3D 数据曲面的绘制和 3D 函数曲面的绘制基本相同，所以我们无需多加解释，直接来看命令和结果：

```
gnuplot> set xlabel "X"
gnuplot> set ylabel "Y"
gnuplot> set zlabel "Z"
gnuplot> set xyplane 0.2
gnuplot> set hidden3d
gnuplot> set view 45,30
gnuplot> unset key
gnuplot> splot "data3d.dat" with lines
```



曲面后面的部分由于遮挡，看不到。没关系，我们学过怎样调整视角：

```
gnuplot> set view 45,210
gnuplot> replot
```

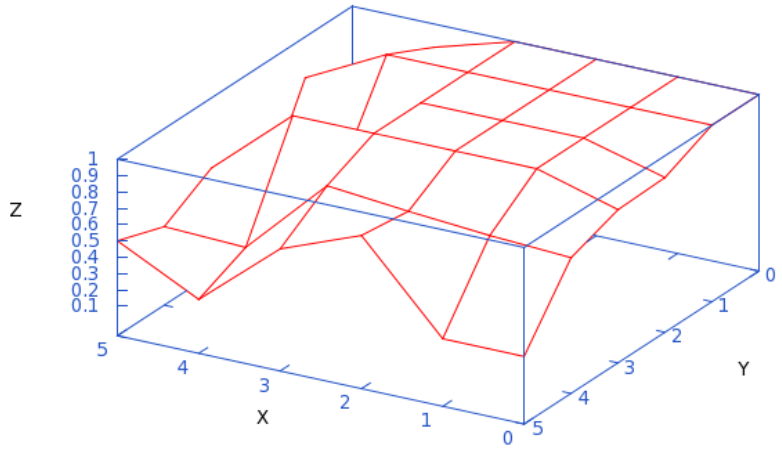


这里还要讲讲边框的问题。在 2D 情况下我们曾经讲过可以通过 unset border 命令取消边框，但是我们没有讲 set border 命令，因为这个只有在 3D 情况下才能说清楚。2D 情况下只有 4 个边框，而 3D 情况下，除了底部 4 个边框，还有竖直 4 个，以及顶部 4 个，所以总共 12 个边框。这 12 个边框显示与否，由一个 12 bit 的整数控制，例如：

- `0000 0000 1111`: 只显示底部 4 个边框
- `0000 1111 0000`: 只显示竖直 4 个边框
- `1111 0000 0000`: 只显示顶部 4 个边框

不难看出，如果要显示所有 12 个边框，对应的二进制数就是：`1111 1111 1111`，换算成十进制就是 4095。set border 命令之后紧跟这个整数，就可以控制显示哪些边框。另外，以前曾经讲过的和 linestyle 有关的参数，都可以用来控制边框的显示风格：

```
gnuplot> set border 4095 lc rgb "#2554C7"
gnuplot> replot
```

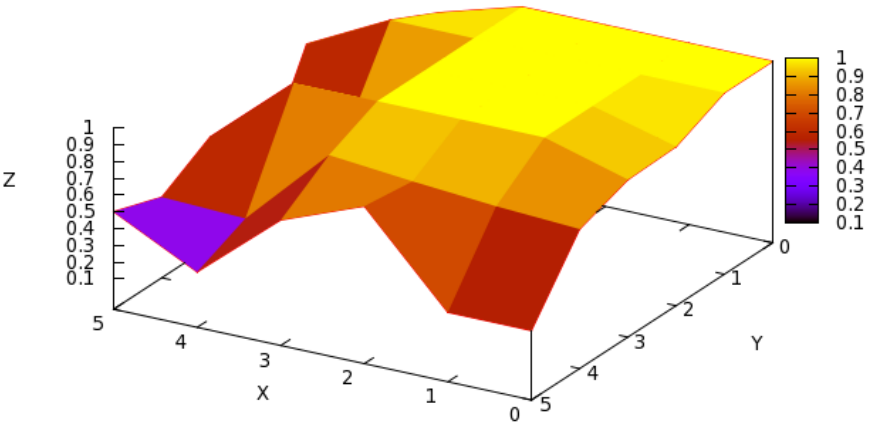


28 Pm3d 绘图

在 3D 函数绘图中我们介绍过，使用 pm3d 模式可以用色彩表示 3D 函数值的大小。数据绘图也可以使用 pm3d 模式。我们这次就稍微详细介绍一下 pm3d。

pm3d 即 Palette Map 3D，中文或叫“色板映射”。我们还是拿上一篇文章中的数据文件来举例子：

```
gnuplot> set xlabel "X"
gnuplot> set ylabel "Y"
gnuplot> set zlabel "Z"
gnuplot> set xyplane 0.2
gnuplot> unset key
gnuplot> set hidden3d
gnuplot> set pm3d
gnuplot> set view 45,210
gnuplot> splot "data3d.dat" with lines
```



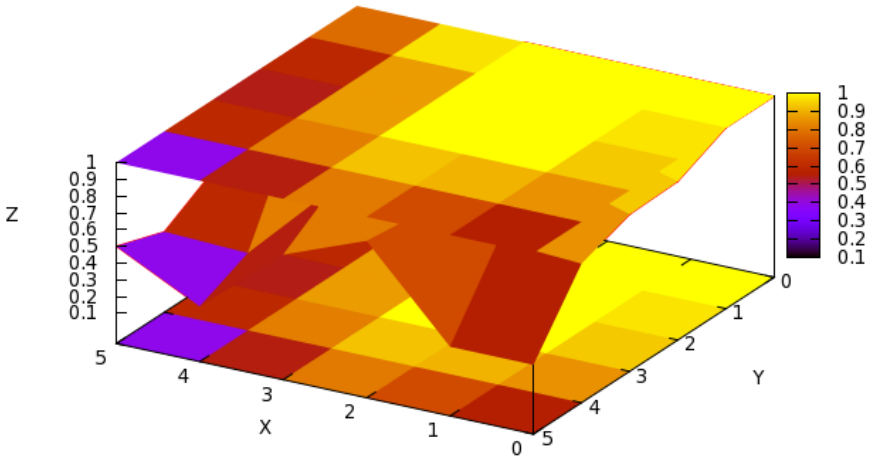
我们看到，本来的曲面被加上了颜色，不同的色彩对应不同的 z 值，这个对应关系显示在右边的色彩条（color box）中。这里有个问题：我们的数据是 6x6 的，但是色彩块只有 5x5，这个色彩是怎样确定的呢？很简单，每个块的四个顶点取平均值，对应的色彩就是这个块的色彩。这个取值方法不是唯一的，可以用 corners2color 参数来设定，详情请用 help set pm3d 命令查询。

色彩图除了画在曲面上，还可以画在底部或顶部（还记得 3D 图底部和顶部都有 border 吗？）设置方法为：

```
set pm3d at b|s|t
```

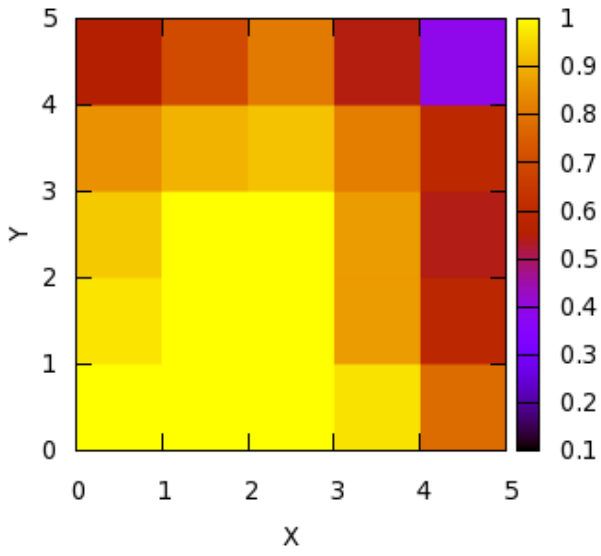
b,s,t 三个字母分别代表底部、曲面和顶部，at 之后可以是任一个字母，也可是三个字母的任意组合。例如：

```
gnuplot> set pm3d at bst
gnuplot> replot
```



当然，如果要看数据在取值范围内的全貌，最好的方法是从上往下看：

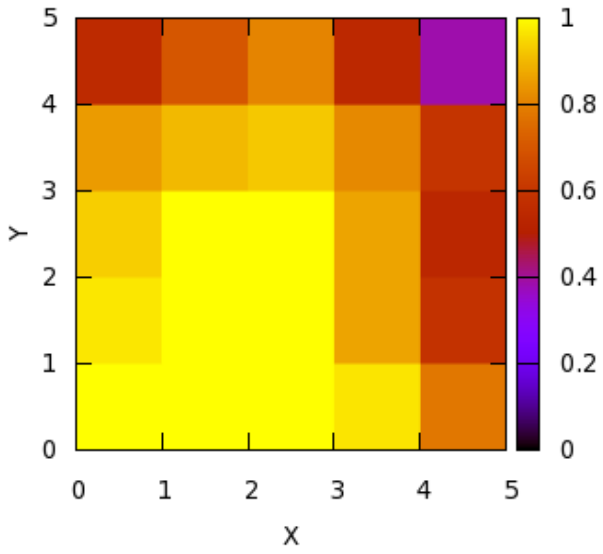
```
gnuplot> set pm3d map
gnuplot> set size square
gnuplot> replot
```



我们之前介绍过 set view 命令调整视角，这里的 set pm3d map 其实是包含 set view map 命令在内的一系列命令组合。

我们注意到，右边的色彩条取值范围是 [0.1:1]，这是 gnuplot 自动设置的。如果我们想手动设置这个范围呢？可能有人会想到 set zrange，但是 zrange 控制的是画图时 z 的取值范围，而不是色板对应的取值范围。其实，色板对应的取值范围是 cbrange（Color Box range）：

```
gnuplot> set cbrange [0:1]
gnuplot> replot
```



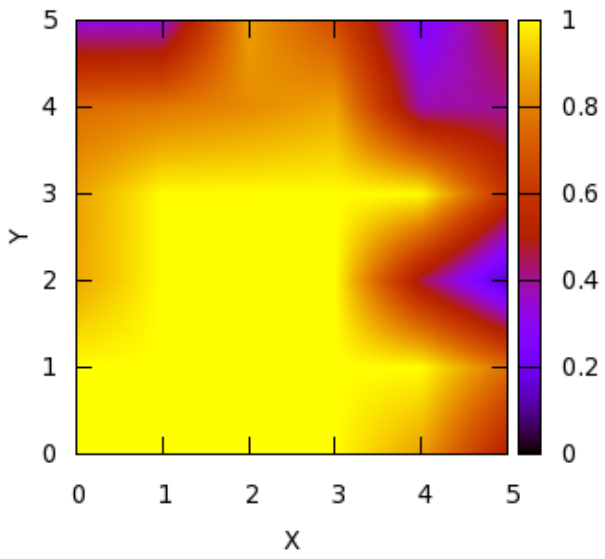
我们什么时候会想要手动设置 cbrange 呢？例如，我们需要比较两组数据的时候，我们一定希望两张图上，相同数值对应的是相同的颜色。

最后，我们有时候可能觉得这样像“马赛克”一样的图片不那么好看。这时候，我们可以利用插值，得到比较平滑的彩色图。使用插值的方法是：

```
set pm3d interpolate N,M
```

M 和 N 分别代表 x 和 y 方向插值的数目。如果希望 gnuplot 自动优化选择，就让 M=N=0：

```
gnuplot> set pm3d interpolate 0,0
gnuplot> replot
```



29 色板（palette）设置

pm3d 绘图默认的色板看起来很不错，但是我们有时候还是希望能自己定义不同的色彩，这次我们谈谈怎样自定义色板。

自定义色板的方式有好多种，我们这里只谈一下比较方便常用的方式：用 rgbformulae 定义 RGB 色彩。

RGB 是电脑中最常用的色彩空间表示方式，而 rgbformulae 是一系列从数值到色阶的数学映射公式，共有 37 个。如果想知道 rgbformulae 到底包含哪些公式，可以使用 gnuplot 命令：

```
show palette rgbformulae
```

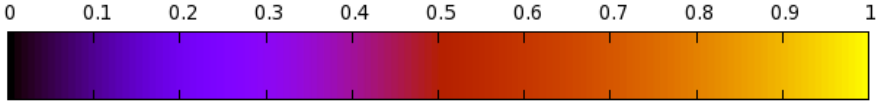
由于 RGB 有三个颜色通道，所以每一个色板需要三个公式，分别表示 R（Red）、G（Green）、B（Blue）。色板的设置方法为：

```
set palette rgbformulae r,g,b
```

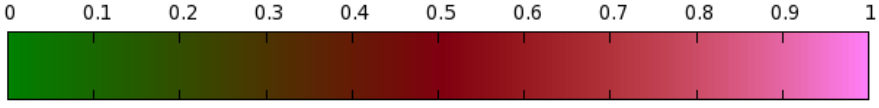
其中 r, g, b 分别表示 R, G, B 通道所用公式代码（0 到 36，允许用负值）。gnuplot 默认色板的公式代码为 7，5，15。

37 个公式加上负值共有 73^3 个不同的组合方式，并不是每个组合都有好的显示效果，而我们仅凭想象力是无法知道效果好坏的。这时候，test palette 命令可以帮助我们预览一下显示效果。即使如此，找到一个好的色板设置组合并不是一件很容易的事情。下面这些常用的组合可以给我们提供一些帮助：

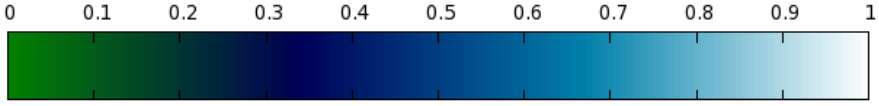
1. 7, 5, 15（pm3d 默认）



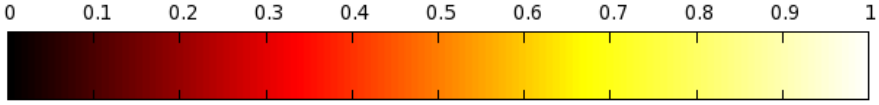
2. 3, 11, 6（绿—红—紫）



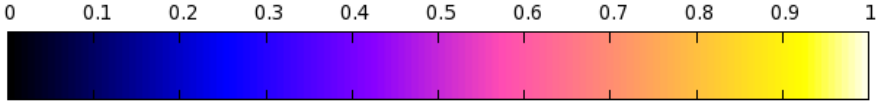
3. 23, 28, 3（绿—蓝—白）



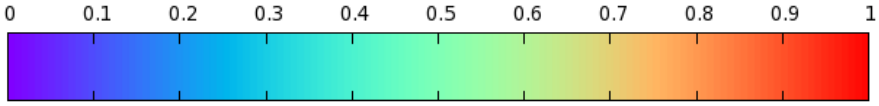
4. 21, 22, 23（黑—红—黄—白）



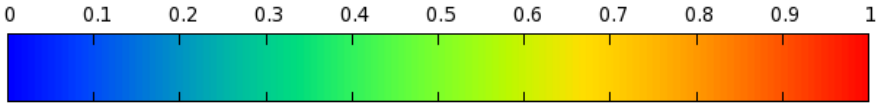
5. 30, 31, 32（黑—蓝—紫—黄—白，可以用于黑白打印）



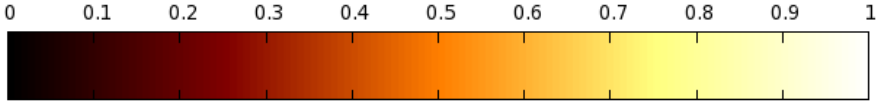
6. 33, 13, 10（彩虹色）



7. 22, 13, -31（另一种彩虹色）



8. 34, 35, 36（黑—红—黄—白）



最后，如果我们不想用彩色，而只是想用黑白灰度，可以用下面的命令：

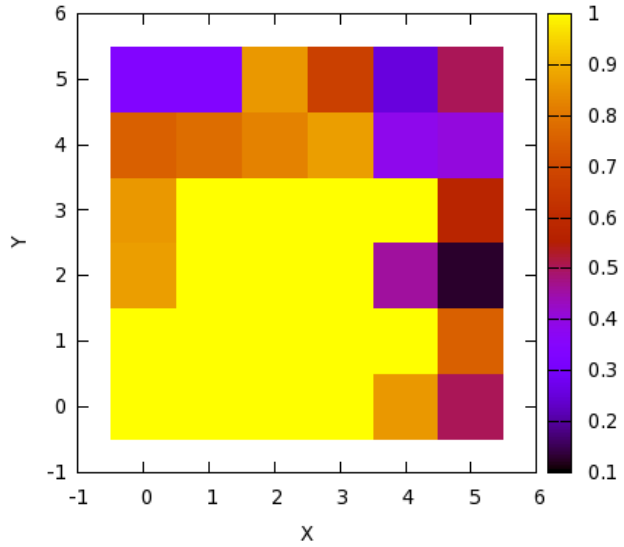
```
set palette gray
```

30 Image 绘图

在谈到 pm3d 绘图时我们说过，NxM 的数据只能画出 (N-1)x(M-1) 的图像。有没有这样一种方式，让我们能从 NxM 的数据画出 NxM 的图像呢？这次我们介绍一种新的画图风格：image。

咱们还是用以前用过的数据文件 data3d.dat：

```
gnuplot> unset key
gnuplot> set xlabel "X"
gnuplot> set ylabel "Y"
gnuplot> set size square
gnuplot> plot "data3d.dat" with image
```



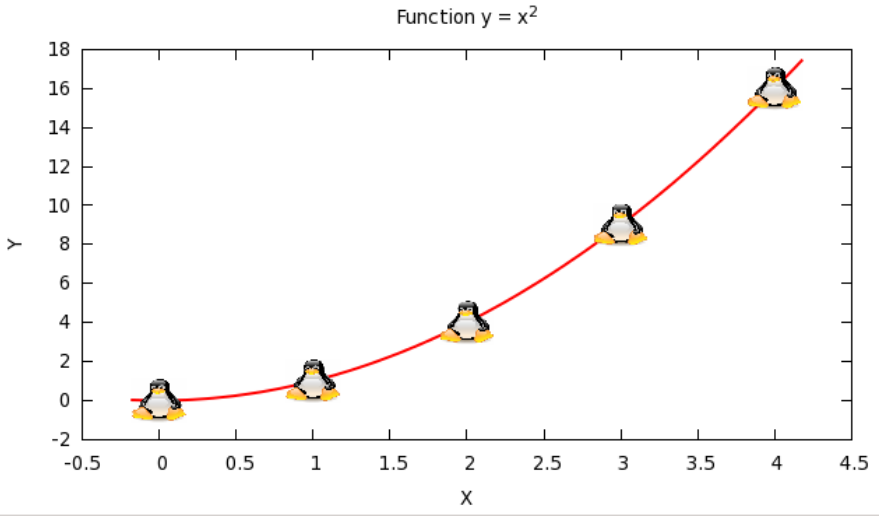
这里我们看到，色块数目等于数据点数目，每一个色块中心位于相应数据点，而色块色彩对应的就是该数据点的值。这和 pm3d 不同，因为 pm3d 绘图时数据点位于色块顶点位置。另外，这里虽然用的是 plot 命令，但是图像反应了 3D 的信息，所以我们把它放在 3D 作图里介绍。

这两种方式无法简单判断优劣，只能根据实际需要选择。当像素比较多而数据又比较平滑的时候，其实两者差别不大。

除了 image，还有两个非常类似的绘图方式：rgbimage 和 rgbalpha。image 用于处理单色图（只有一个 z 值），可以包含 x,y,z 三列数据；rgbimage 用于处理 RGB 彩色图，可以包含 x,y,r,g,b 五列数据；rgbalpha 在 rgbimage 基础上增加了 alpha 通道（透明信息），可以处理透明 RGB 彩色图。

除了一般数据文件，上述画图方式还支持直接读取 png 格式图片：

```
gnuplot> unset key
gnuplot> set title "Function y = x^2"
gnuplot> set xlabel "X"
gnuplot> set ylabel "Y"
gnuplot> set term wxt enhanced
gnuplot> plot x**2 lw 2,\
> "tux.png" binary filetype=png center=(0,0) dx=0.002 dy=0.01 with rgbalpha,\
> "tux.png" binary filetype=png center=(1,1) dx=0.002 dy=0.01 with rgbalpha,\
> "tux.png" binary filetype=png center=(2,4) dx=0.002 dy=0.01 with rgbalpha,\
> "tux.png" binary filetype=png center=(3,9) dx=0.002 dy=0.01 with rgbalpha,\
> "tux.png" binary filetype=png center=(4,16) dx=0.002 dy=0.01 with rgbalpha
```



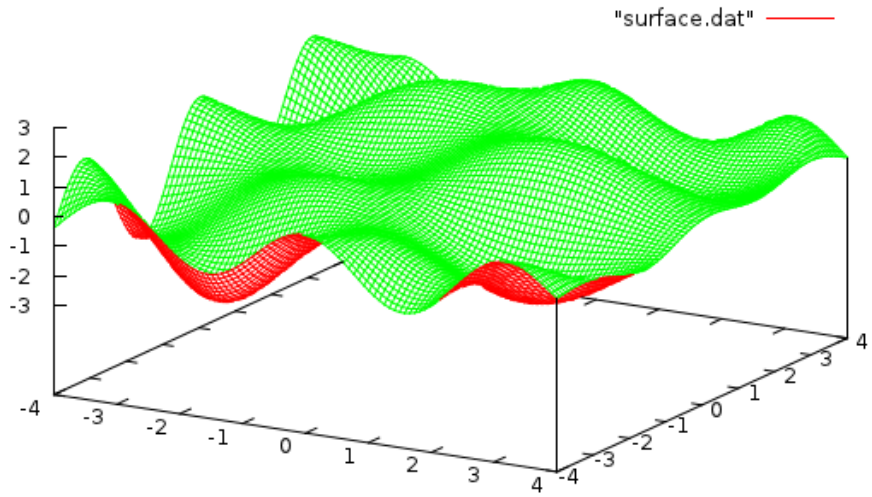
这里我们利用一个小企鹅的图片（tux.png）画了一个二次平方函数。binary 文件格式我们以前介绍过，这里有两个新的参数：filetype 和 center。filetype 用于指定文件格式，而 center 用于指定图片中心位置。

31 等高线图

等高线图是另一类重要的 3D 绘图方式。为了说明 gnuplot 里面等高线图的绘制方法，我们使用下面这个数据文件作为例子：surface.dat

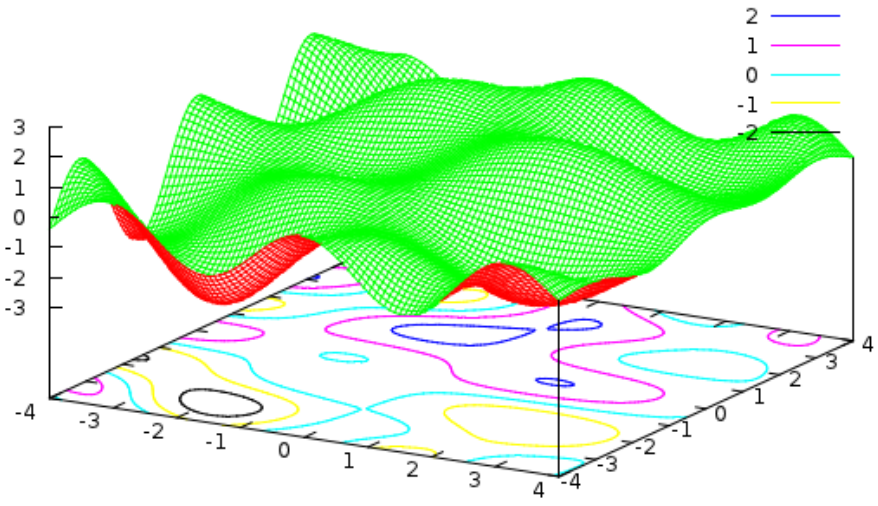
首先绘制普通曲面图：

```
gnuplot> set hidden3d
gnuplot> splot "surface.dat" with lines
```



下面加上等高线：

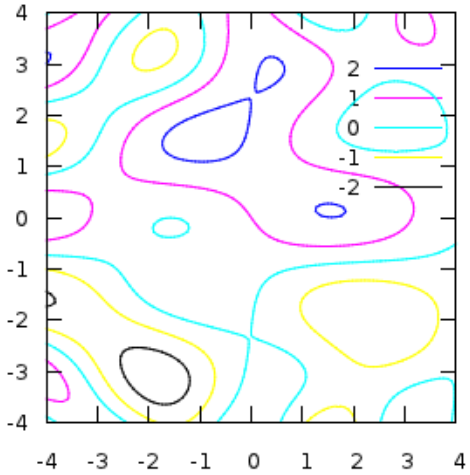
```
gnuplot> set contour base
gnuplot> splot "surface.dat" with lines title ""
```



set contour 命令之后除了 base 参数外，还可以使用 surface 或 both 参数，分别表示等高线画在底面、曲面或者两者都画。这里设置了一个空的 title，是为了在图例中不要显示文件名，以免和等高线的图例混淆。

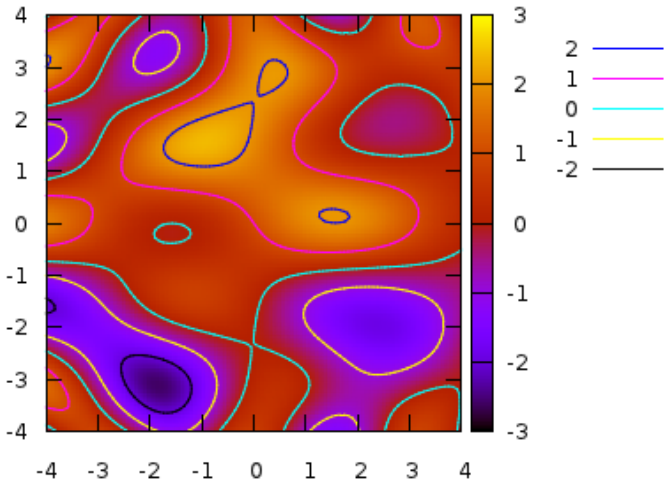
如果我们想在平面中显示等高线，可以使用下列命令：

```
gnuplot> unset surface
gnuplot> set view map
gnuplot> set size square
gnuplot> replot
```



如果我们想在之前提到过的 pm3d 图上显示等高线，可以这样做：

```
gnuplot> set pm3d at b
gnuplot> set key at screen 0.8,0.8
gnuplot> replot
```



这里我们把图例的位置做了调整，因为默认图例是在图像里面的，这样可能影响我们的图像显示。

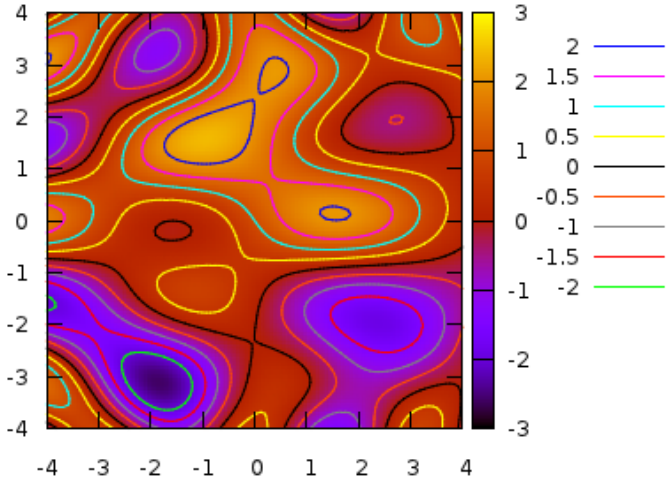
最后，我们谈谈怎样手动设置等高线数值和间距。等高线的数值间隔参数设置命令是 set cntrparam levels。默认情况下，gnuplot 自动设置等高线数值。如果要进行手动设置，有两种方法:

1. `incremental <start>,<incr>,<end>`
   设置起始值以及间隔大小，这种方法适用于等间隔的等高线;

2. `discrete <z1>,<z2>,<z3>,...`
   分别设置各个等高线数值，这种方法适用于间隔不等的等高线。

例子:

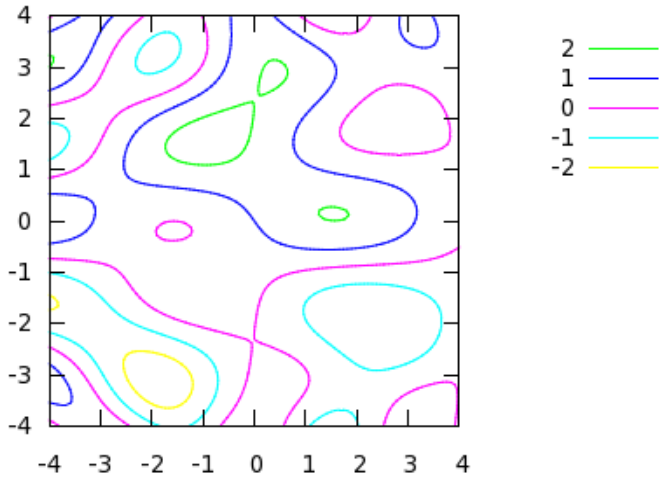
```
gnuplot> set cntrparam levels incremental -2,0.5,2
gnuplot> replot
```



32 等高线的颜色

上一讲最后我们提到了怎样改变等高线的数值间距。不同等高线是由颜色来区分的，而默认的颜色未必是最理想的组合，我们还用上次的数据文件举例:

```
gnuplot> set contour
gnuplot> unset surface
gnuplot> set size square
gnuplot> set key at screen 0.8,0.8
gnuplot> set view map
gnuplot> splot "surface.dat" with lines title ""
```

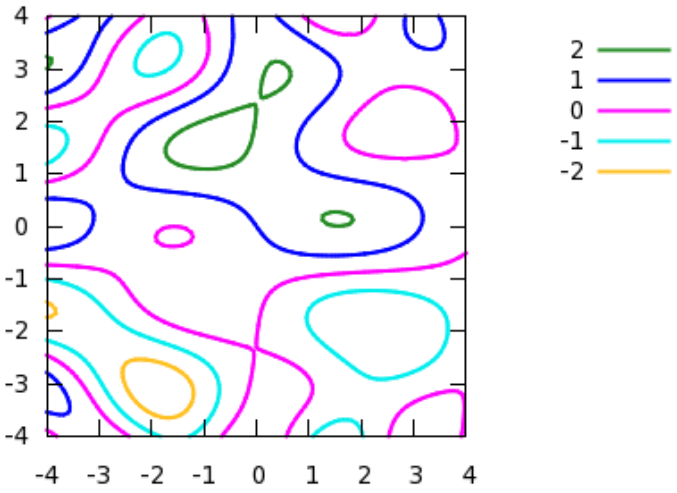


这里有些默认的颜色太浅了，看不清楚。下面我们想办法把它们加深加粗。

细心的读者可能注意到，这里默认的等高线颜色和上一讲中的不同。等高线的默认颜色是这样确定的：按照色彩编号（如果不记得了，请复习一下我们以前讲过的“点线风格”）从低到高，而最低的色彩比曲面（surface）颜色大一号。这里虽然曲面没有画出来，但是仍然占去了 1 号颜色，所以等高线颜色从 2 号颜色（绿色）开始。而上一讲中，我们设置了 hidden3d 参数，曲面上下面分别使用不同的颜色，因此曲面占去 2 个颜色，等高线从 3 号颜色（蓝色）开始。

我们现在要做的就是要 gnuplot 使用我们自定义的颜色，而不是系统默认的颜色。下面看代码：

```
gnuplot> set style line 1 lw 2 lc rgb "red"
gnuplot> set style line 2 lw 2 lc rgb "forest-green"
gnuplot> set style line 3 lw 2 lc rgb "blue"
gnuplot> set style line 4 lw 2 lc rgb "magenta"
gnuplot> set style line 5 lw 2 lc rgb "dark-cyan"
gnuplot> set style line 6 lw 2 lc rgb "goldenrod"
gnuplot> set style increment userstyles
gnuplot> replot
```



我们这里首先自定义了一系列 linestyle（参考第十八讲“图例”），然后用 set style increment userstyles 命令告诉 gnuplot 使用我们自己定义的曲线风格，而不是默认风格。这样，我们得到了我们想要的效果。

这里我们并没有使用 RGB 直接定义颜色，而是用了一系列色彩的名字（colornames），这些 colornames 是 gnuplot 预定义的一些 RGB 颜色。如果想知道到底有哪些预定义的颜色，可以使用下面的命令：

```
gnuplot> show palette colornames
```

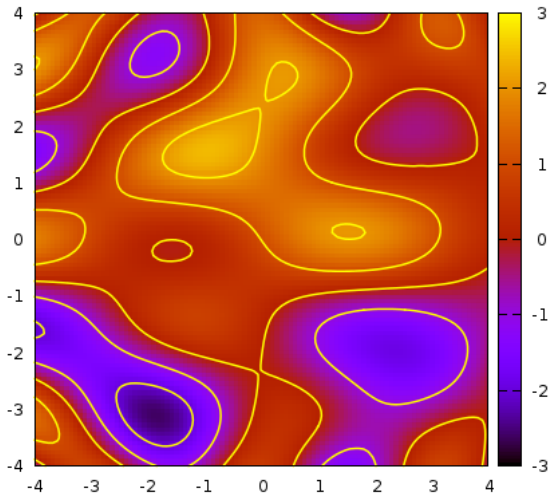
其实不仅限于等高线，只要是在同一个图上画多个函数或数据，都可以用这种方法设置显示风格。

33 Table 输出

gnuplot 作为一个绘图软件，一般都是读取数据，把图像输出到指定的 terminal。但是 gnuplot 也可以把图像以 table 形式存储到一个数据文件中。这样做的好处是方便我们对图像数据进行进一步处理，得到我们想要的效果。我们之前画等高线图所用的 surface.dat 文件，其实就是通过这种方式得到的。

还以 surface.dat 数据的等高线图为例，contour 模式只能在 splot 命令中使用。如果我们把等高线图预先以 table 格式存入一个文件，我们就可以对其进行适当处理，并且可以用 plot 命令绘图：

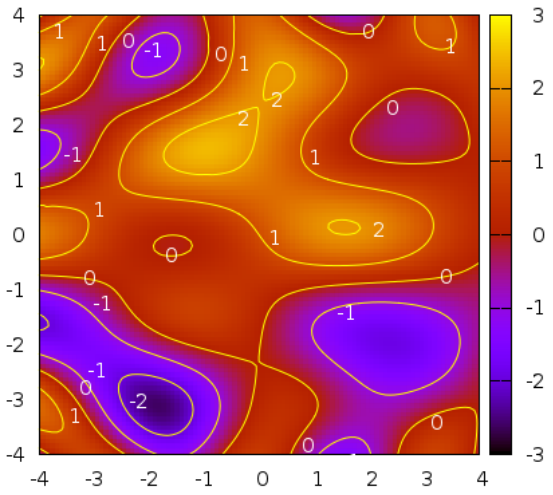
```
gnuplot> set contour base
gnuplot> unset surface
gnuplot> set table 'cont.dat'
gnuplot> splot 'surface.dat'
gnuplot> unset table
gnuplot> unset key
gnuplot> set size square
gnuplot> set xrange [-4:4]
gnuplot> set yrange [-4:4]
gnuplot> plot 'surface.dat' with image, 'cont.dat' with lines lw 1.5 lc rgb '
    yellow'
```



这里我们预先把等高线图以 table 格式存入了 cont.dat 文件，随后用 plot 命令调用该文件，把等高线画在了 surface.dat 数据的 image 图上（注意：不是 pm3d 图）。这里我们可以用 plot 命令的图线风格参数控制等高线的粗细和颜色。

这里我们没有像以前那样用不同颜色区分等高线，而是所有等高线使用了同一颜色。那么怎样区分各个等高线呢？我们可以用 label 标注等高线数值。label 的使用方法我们在第十五讲“坐标系统及标签”里介绍过，但是直接手动设置 label 有点麻烦。下面这个网页里的 label_contours.awk 文件是一个 AWK 脚本文件，可以帮助我们自动设置标签： http://gnuplot.sourceforge.net/scripts/index.html#tricks-here

我们这里不打算介绍 AWK 脚本语言，也不打算介绍这个脚本的具体使用方法，因为该网页以及该文件内已经包含了详细的使用说明。我们只给出一个使用该脚本得到的带标注的等高线图：

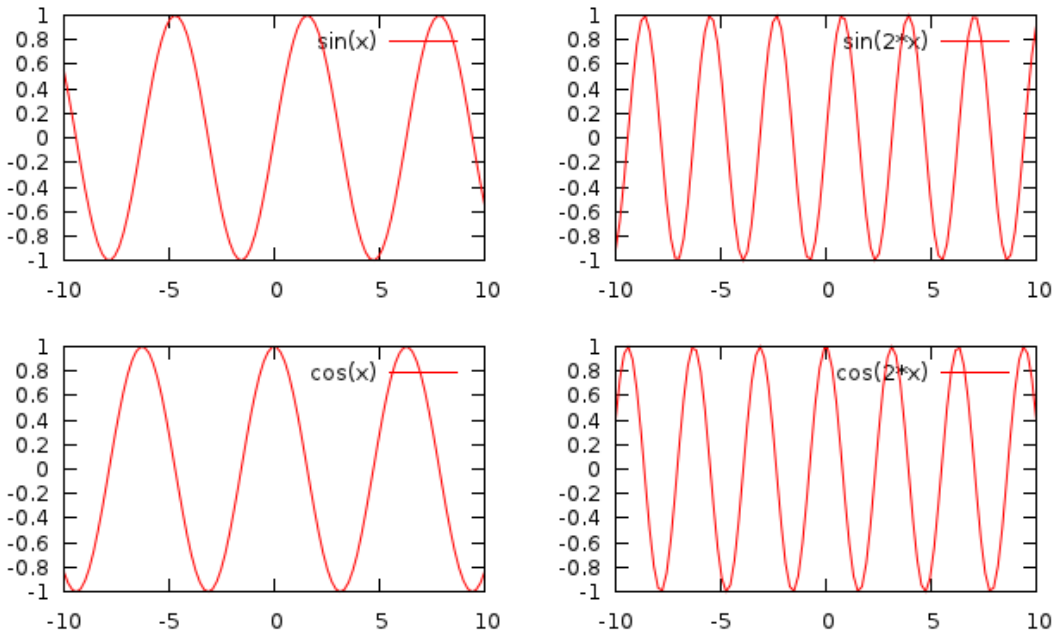


34 多图（multiplot）

之前我们曾用 plot 命令将多个曲线画在同一张图上，这次我们谈谈怎样将多个图画在一起。这样做的主要目的是把不同的信息放在一起，以方便比较。我们首先看看怎样把几张图像矩阵一样排列起来：

```
gnuplot> set multiplot layout 2,2
```

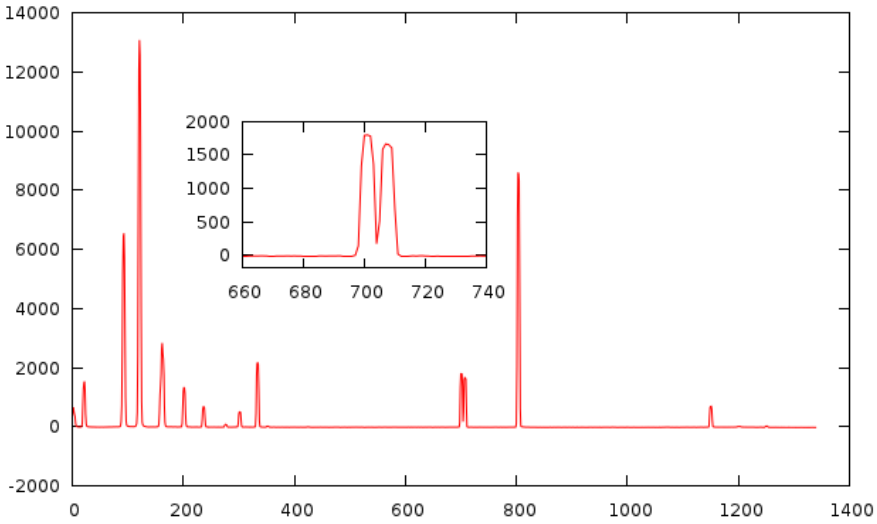
```
gnuplot> plot sin(x)
gnuplot> plot sin(2*x)
gnuplot> plot cos(x)
gnuplot> plot cos(2*x)
gnuplot> unset multiplot
```



set multiplot 命令告诉 gnuplot 进入多图模式，这时候我们画的所有图像都会在出现在同一张画布上。如果不做调整，所有图像将重叠在一起；如果适当调整每个图像的位置和大小，图像将按我们的要求排列起来。layout 参数可以自动将几幅图按照指定格式排列起来，例如这里的 4 幅图排成了 2×2 的格式。最后，unset multiplot 命令退出多图模式。之前我们用过的绘图命令在多图模式下都有效，包括将图片输出为不同格式。

如果不用 layout 参数，我们还有更灵活的排版方式，例如下面这个“画中画”的例子：

```
gnuplot> reset
gnuplot> set multiplot
gnuplot> unset key
gnuplot> plot 'spectrum.dat' with lines
gnuplot> set xrange [660:740]
gnuplot> set xtics 20
gnuplot> set yrange [-200:2000]
gnuplot> set ytics 500
gnuplot> set origin 0.2,0.4
gnuplot> set size 0.4,0.4
gnuplot> replot
gnuplot> unset multiplot
```



这里我们绘制了一组光谱数据（spectrum.dat），并且把 [660, 740] 这个区间内的数据单独绘制出来，嵌套在大图内，用于更清楚的展示细节。这里的 set origin 命令用于设置绘图的原点，set size 命令以前已经讲过，用于设置图像大小。这两个命令均使用 screen 坐标系统（参考之前的博文：“坐标系统及标签”）。开始处的 reset 命令正如它的名字所暗示的那样，用于让人工设置的参数恢复至默认值。

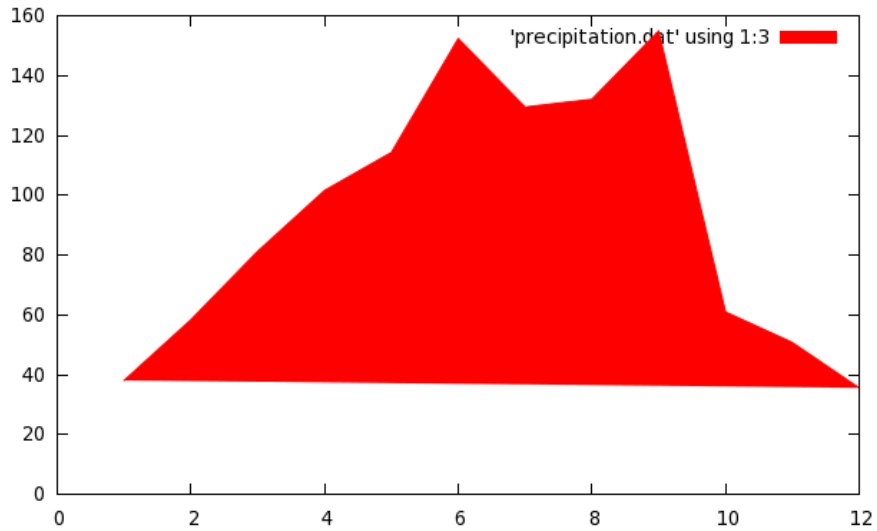
如果想更多了解 multiplot 命令的用法，请使用 help multiplot 命令。

35 曲线色彩填充

目前我们所涉及的主题都是比较基本的，应该覆盖了大部分科学绘图的需要，而 gnuplot 的功能还远不止如此。从现在开始，我们介绍一些稍微高级些的应用，其中有些可以让我们的绘图更加丰富多彩，有些能让我们的工作更有效率。

这次我们介绍一下填充曲线（filledcurves）。我们使用在“多组数据绘图”中用过的数据文件“precipitation.dat” 作为例子：

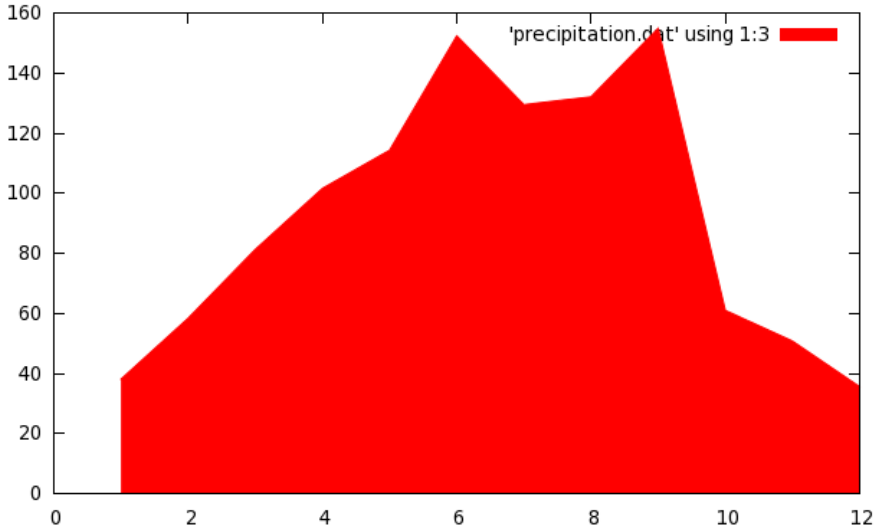
```
gnuplot> plot 'precipitation.dat' using 1:3 with filledcurves
```



这里的 filledcurves 是一种新的画图风格，它会在数据（或函数）曲线中填入指定颜色。默认情况下，曲线是自我封闭的（closed），所以这里的例子中首尾数据点被连接了起来。

我们也可以指定填充曲线和某条水平或竖直直线之间的区域：

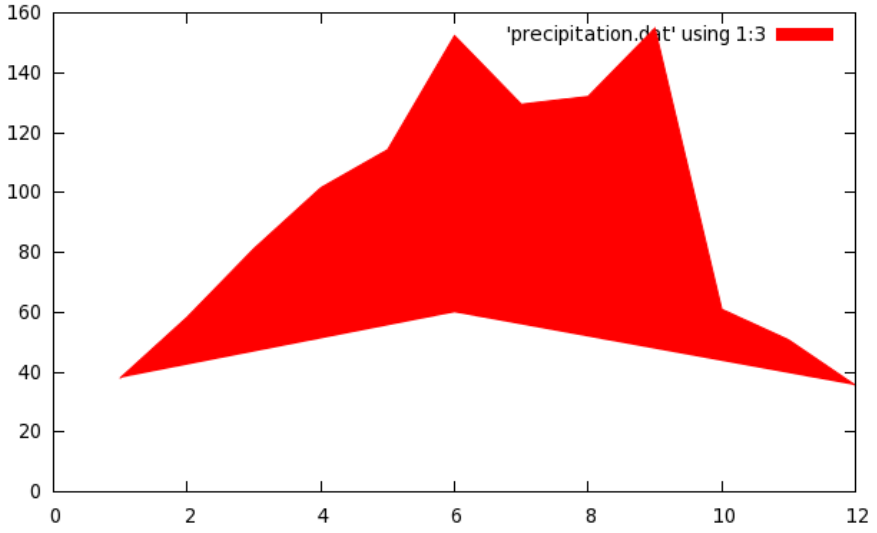
```
gnuplot> plot 'precipitation.dat' using 1:3 with filledcurves y1=0
```



这里曲线和横轴之间的区域被填充了颜色。这里的 y1 也可以是 x1, x2, y2，它们所代表的意义在“第二坐标轴”中介绍过。

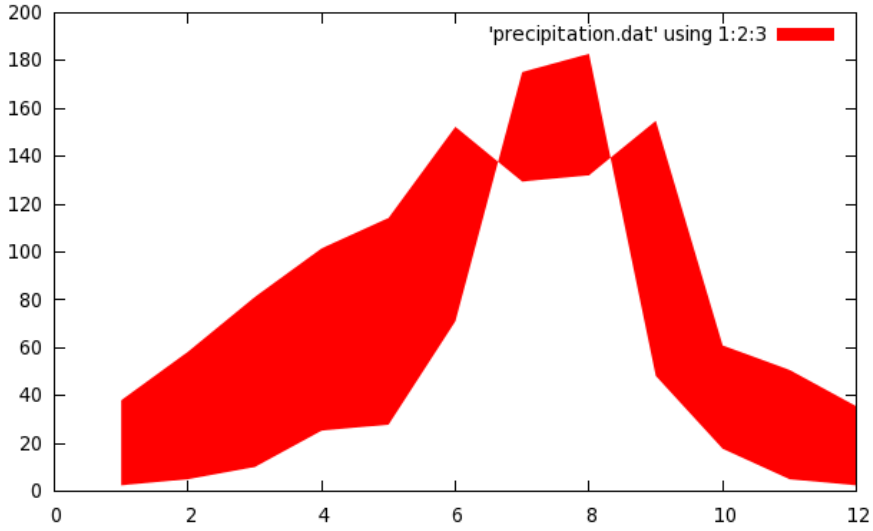
除此之外，也可以指定曲线和某点之间的连接区域：

```
gnuplot> plot 'precipitation.dat' using 1:3 with filledcurves xy=6,60
```



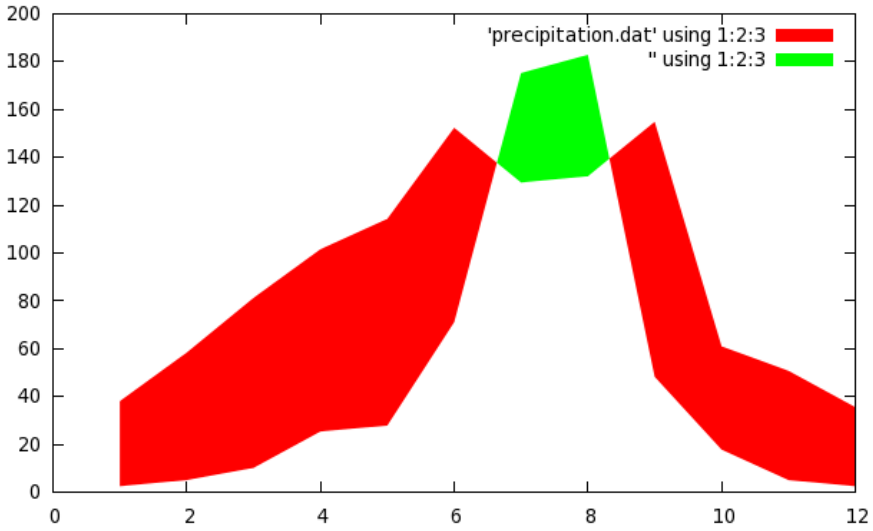
如果提供两组数据，还可以在两组数据之间的区域填充颜色：

```
gnuplot> plot 'precipitation.dat' using 1:2:3 with filledcurves
```



如果我们希望当第一组数据小于第二组时使用一种颜色，而第一组数据大于第二组时使用另一种颜色，我们可以这样做：

```
gnuplot> plot 'precipitation.dat' using 1:2:3 with filledcurves below, '' using
    1:2:3 with filledcurves above
```



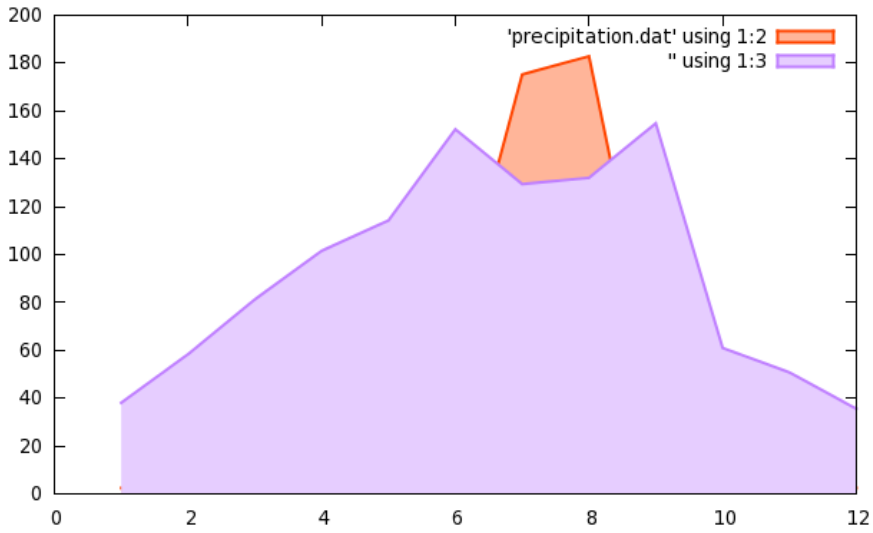
这里的 below 和 above 参数表示只填充相应（低于或高于）的区域。plot 命令第二部分的数据文件名为空，这是因为该文件和第一部分是相同的，我们可以把文件名省略。

我们虽然得到了具有填充色的曲线，但是这样大红大绿的效果并不是我们想要的。我们下一次就讲讲怎样设置填充风格，让它们看起来更好看一点。

36 填充风格

正如曲线风格的设置命令是 set style line，色彩填充风格的设置命令是 set style fill。我们仍然以 “precipitation.dat” 文件举例：

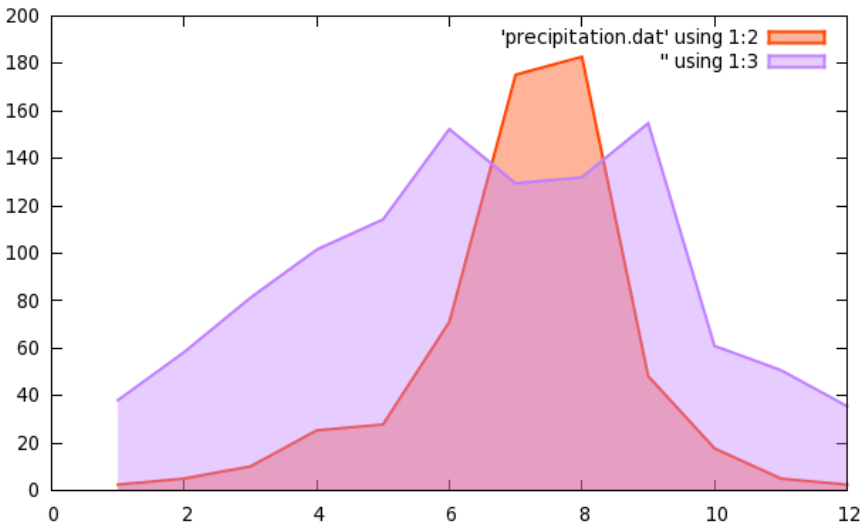
```
gnuplot> set style fill solid 0.4
gnuplot> plot 'precipitation.dat' using 1:2 with filledcurves y1=0 lw 2 lc rgb
   "orange-red", '' using 1:3 with filledcurves y1=0 lw 2 lc rgb "purple"
```



这里的 solid 参数表示用纯色填充，后面的数字取值范围从 0 到 1，表示色彩深浅。默认情况下，填充色和曲线颜色是相同的。

这里有个问题：第一组数据有一部分被第二组数据覆盖住了。有些 terminal（例如wxt, png, pdf 等）支持透明效果，可以解决这个问题。

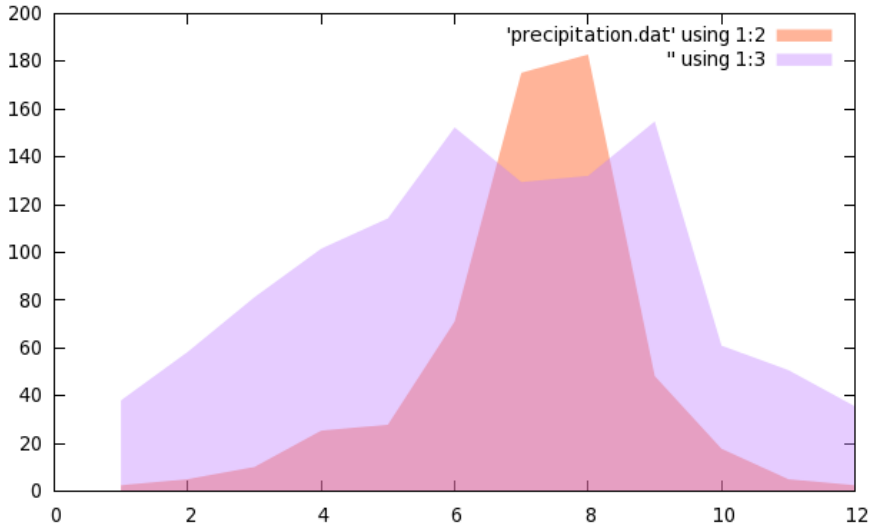
```
gnuplot> set style fill transparent solid 0.4
gnuplot> replot
```



这里使用了 transparent 参数，相应的 solid 后面的数字就变成了表示透明度（alpha 值）的参数。

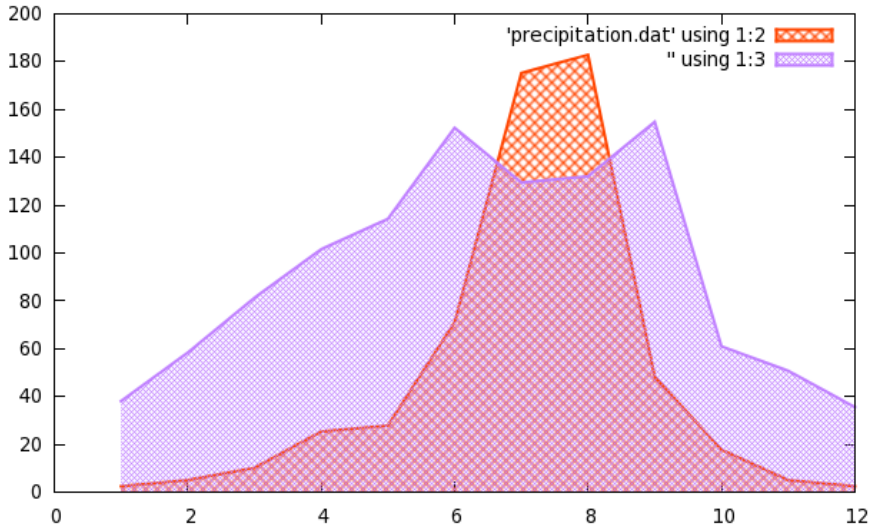
如果我们希望填充区域不要有边线（曲线本身），那么可以使用 noborder 参数：

```
gnuplot> set style fill transparent solid 0.4 noborder
gnuplot> replot
```



如果需要黑白打印，我们可能希望不要用纯色填充，而是用不同的花纹填充，这时我们可以使用 pattern 参数。pattern 参数之后可以跟一个数字，表示起始使用的 pattern 编号。不同的 terminal 有不同的花纹效果，可以在某 terminal 下使用 test 命令查看该 terminal 支持哪些花纹。下面是例子：

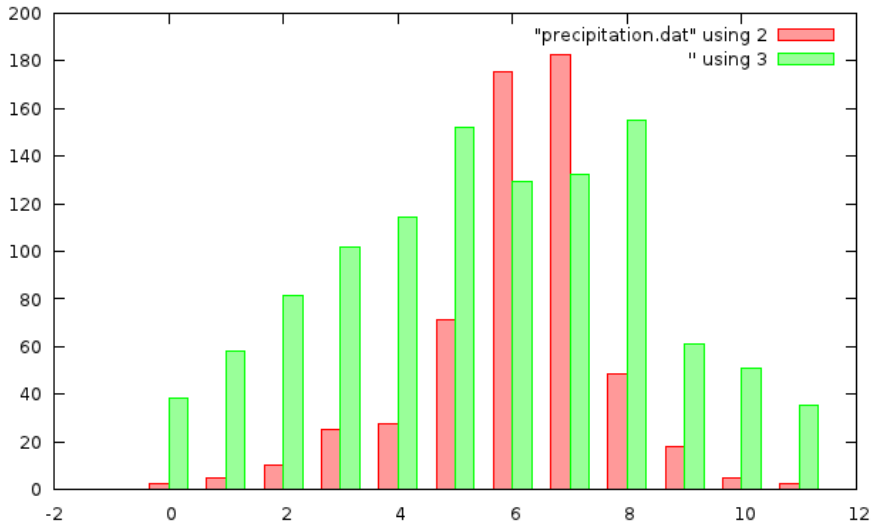
```
gnuplot> set style fill transparent pattern 1 border
gnuplot> replot
```



37 柱状图

除了点线图外，柱状图也是一种常用的作图方式。我们还用前两讲用过的 “precipitation.dat” 数据文件作例子：

```
gnuplot> set style data histogram
gnuplot> set style histogram clustered gap 1
gnuplot> set style fill solid 0.4 border
gnuplot> plot "precipitation.dat" using 2, '' using 3
```



set style data histogram 命令告诉 gnuplot 所有数据绘图都使用 histogram 风格。set style histogram 命令设置 histogram 作图的参数，例如 clustered 模式就是像上图那样，把几组数据并排画在一起，gap 1 就表示各簇数据之间空白的宽度等于数据柱宽度的 1 倍。填充风格命令（set style fill）上次我们已经介绍过了。

这里我们发现一个 histogram 和一般点线作图的不同：一般点线作图，每个图都要提供 x,y 两组数据，而 histogram 图只需要提供一组数据，每个数据自动画在 X 轴上的非负整数位置。这时候，X 轴上的标注并不是我们想要的，图例中的标注也不符合我们的要求。我们以前在“坐标取值范围及刻度”和“多组数据绘图”讲过，这两处的标注，可以分别用 set xtics 命令和 title 参数来设置。这里提供另一种改变标注的方法，我们把 “precipitation.dat” 文件稍微修改一下：

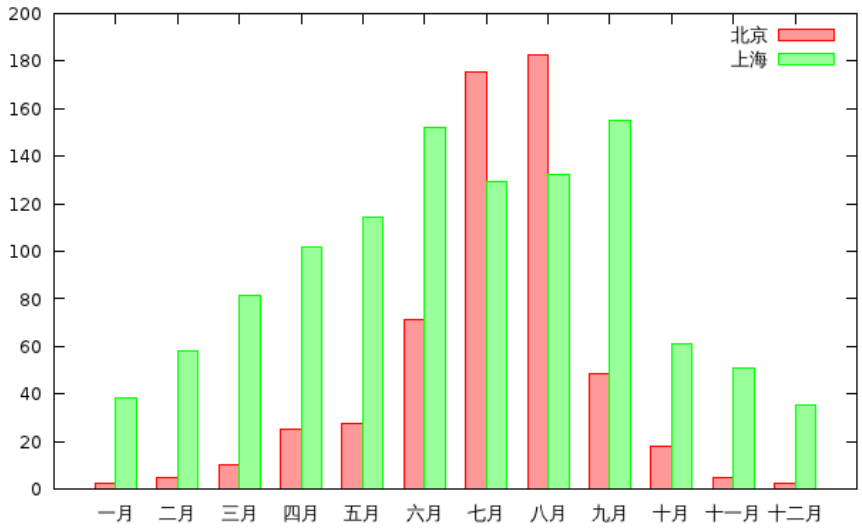
```
### 文件开始 ###
# 各城市月平均降水量 (mm)
#
月份      北京      上海
# ========================
一月      2.5       38.1
二月      5.1       58.4
三月      10.2      81.3
四月      25.4      101.6
五月      27.9      114.3
六月      71.1      152.4
七月      175.3     129.5
八月      182.9     132.1
九月      48.3      154.9
十月      17.8      61.0
十一月    5.1       50.8
十二月    2.5       35.6
### 文件结束 ###
```

我们把修改后的文件称作 “precipitation_v2.dat”。和原文件相比，这里有两处改动：

1. 原文件中包含城市名称的一行是注释掉的，新文件里这一行表示注释的 # 字符被去掉了；
2. 原文件中第一列用阿拉伯数字表示的月份，新文件里被换作了中文月份名称。

下面来看新的例子：

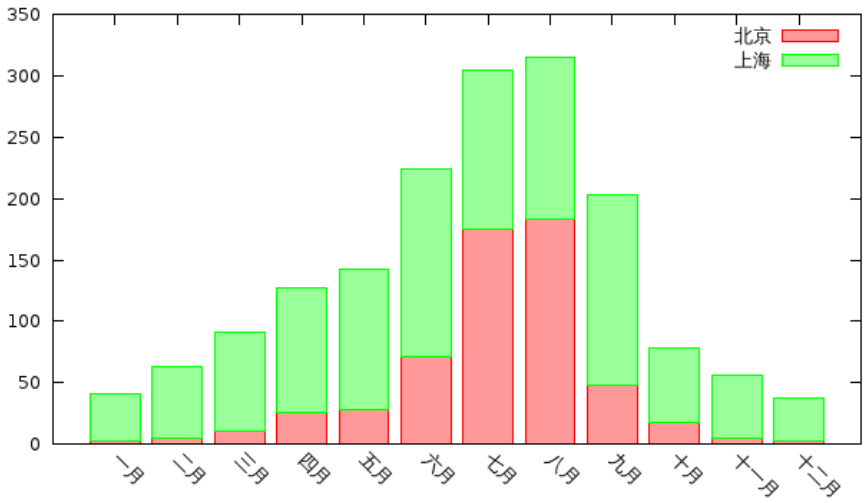
```
gnuplot> plot "precipitation_v2.dat" using 2:xticlabels(1) title columnheader
    (2), '' using 3:xticlabels(1) title columnheader(3)
```



这里 using 参数的数据列号后面，增加了一个新的 xticlabels(1)，这表示使用第 1 列的内容作为每个数据的 X 轴标注。另外，title 参数后面跟了一个 columnheader(n) 函数，这表示使用第 n 列的第一行内容作为每组数据的 title。这样，我们可以把每组数据的横轴和图例标注预先存入数据文件中，画图时再调用。

除了 clustered 模式，histogram 作图还有 rowstacked 模式，就是把几组数据竖着垒起来：

```
gnuplot> set style histogram rowstacked
gnuplot> set boxwidth 0.8 relative
gnuplot> set xtics rotate by -45
gnuplot> replot
```



rowstacked 模式没有 gap 参数，我们这里使用 set boxwidth 命令设置数据柱相对宽度。同时，X 轴标注被旋转了 45 度。这在柱状图里很常见，因为密排的柱状图往往没有足够的空间显示 X 轴标注，旋转之后显示空间便增大了。

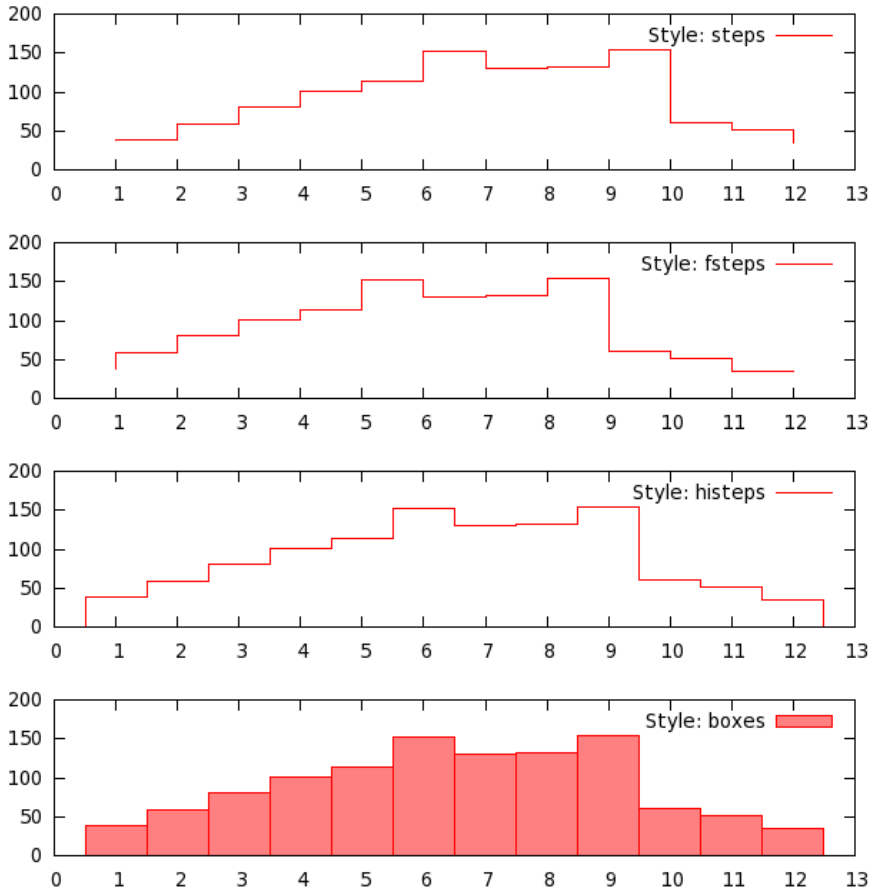
既然有 rowstacked，有没有 columnstacked 呢？答案是有的。rowstacked 逐行把数据叠加显示，而 columnstacked 逐列把数据叠加。除此之外，还有 errorbars 模式，在 clustered 基础上增加误差条。由于这两种模式和之前讲过的很类似，所以就不举例了。

38 阶梯图

和柱状图（histogram）相似的，还有阶梯图，英文叫做 step 或者 stair 图。这种作图方式经常用于绘制数字化的离散变量。gnuplot 里面有三种对应的作图方式，分别为：steps,fsteps,histeps。另外，gnuplot 里面还有一种和 histogram 更为相似的方式：boxes，我们在这里一并介绍。

下面，我们还是用 “precipitation.dat” 数据文件作为例子，来看看这几种作图方式有什么异同：

```
gnuplot> set term wxt size 640,640
gnuplot> set xrange [0:13]
gnuplot> set xtics 1
gnuplot> set ytics 50
gnuplot> set multiplot layout 4,1
gnuplot> set tmargin 1
gnuplot> set style fill solid 0.5
gnuplot> plot 'precipitation.dat' u 1:3 with steps t "Style: steps"
gnuplot> plot 'precipitation.dat' u 1:3 with fsteps t "Style: fsteps"
gnuplot> plot 'precipitation.dat' u 1:3 with histeps t "Style: histeps"
gnuplot> plot 'precipitation.dat' u 1:3 with boxes t "Style: boxes"
gnuplot> unset multiplot
```



我们可以看出，三种 steps 作图方式基本相同，只是在 X 轴方向上略有平移。steps 曲线先沿 X 方向前进，再沿 Y 方向前进；fsteps 曲线先沿 Y 方向前进，再沿 X 方向前进。虽然我们有 12 个数据点，但是 steps 和 fsteps 都只有 11 段表示 Y 值大小的横线。histeps 和 steps 类似，但是起始点前移了 0.5 个单位长度，因此 histeps 具有全部 12 段表示 Y 值大小的横线。这三种 steps 方式都不具有填充风格。boxes 和 histeps 几乎完全相同，不同点在于 boxes 可以使用填充风格。另外，boxes 的柱宽可以通过 set boxwidth 命令调节。

boxes 和 histogram 风格的主要区别在于，histogram 更多用于多组数据的比较，而 boxes 更多用于单组数据的展示。

39 数据平滑

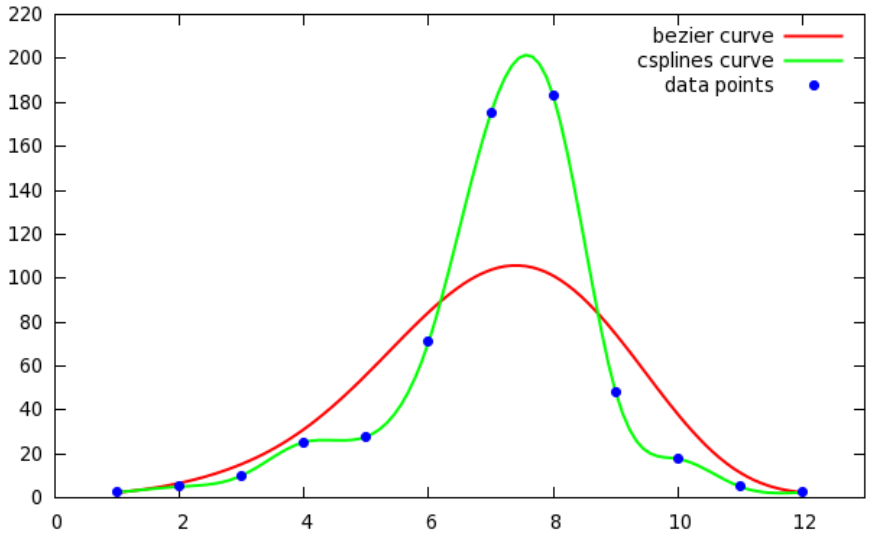
gnuplot 是一个绘图软件，数据处理不是它的长项，但是它也可以做一些简单的数据处理工作，例如之前提到过的拟合。这次我们谈谈另一种常用的数据处理：数据平滑。

数据平滑，就是根据一组数据，绘制出一个比较平滑的曲线。它有很多种不同的算法，一般可分为两类：

1. 根据数据绘制出一个近似的平滑曲线，曲线没必要通过每一个数据点；
2. 曲线通过每一个数据点，在数据点之外通过插值获得平滑的曲线。

gnuplot 里面数据平滑的命令是 smooth，后面紧跟着一个表示具体算法的参数。下面我们来看例子，还是用 “precipitation.dat” 数据文件：

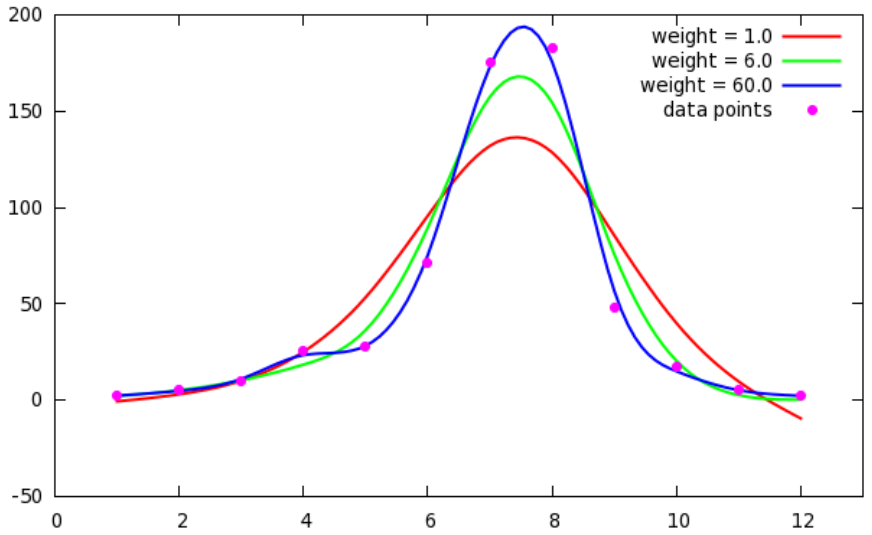
```
gnuplot> set xrange [0:13]
gnuplot> plot 'precipitation.dat' u 1:2 smooth bezier lw 2 t 'bezier curve',\
> '' u 1:2 smooth csplines lw 2 t 'csplines curve',\
> '' u 1:2 w points pt 7 t 'data points'
```



这里有两种平滑算法：bezier 和 csplines，分别对应上述两个类别。smooth 默认使用 lines 风格绘图。

另外，还有一种 acsplines 算法，平滑方式介于上述两者之间。这种算法除了 x 和 y 两组数据外，还要求第三组数据，作为权重（weight）。当然，我们的文件里没有这一组“权重”数据，所以我们手动给所有数据加上相同的权重：

```
gnuplot> plot 'precipitation.dat' u 1:2:(1.0) smooth acsplines lw 2 t 'weight =
    1.0',\
> '' u 1:2:(6.0) smooth acsplines lw 2 t 'weight = 6.0',\
> '' u 1:2:(60.0) smooth acsplines lw 2 t 'weight = 60.0',\
> '' u 1:2 w points pt 7 t 'data points'
```



using 参数里的括号是必要的，否则 gnuplot 会把括号内的数字当成列编号，而不是数字。

这里我们看到，在 acsplines 算法里，数据的权重越小，曲线越平滑，越像 bezier；而数据的权重越大，曲线越接近数据点，也就越像 csplines。

smooth 除了上述算法，还有其他选项，我们下次再介绍。

40 统计直方图

gnuplot 里的 histogram 命令只是一种作图风格，它并不能计算数据的分布并画出直方图。要完成这个任务，我们需要借助 smooth 命令。

上次我们介绍了 smooth 命令的几个参数。其实 smooth 命令不仅可以做数据平滑，也可以进行一些统计计算。这次我们介绍 smooth 的另外两个参数：frequency 和 cumulative。

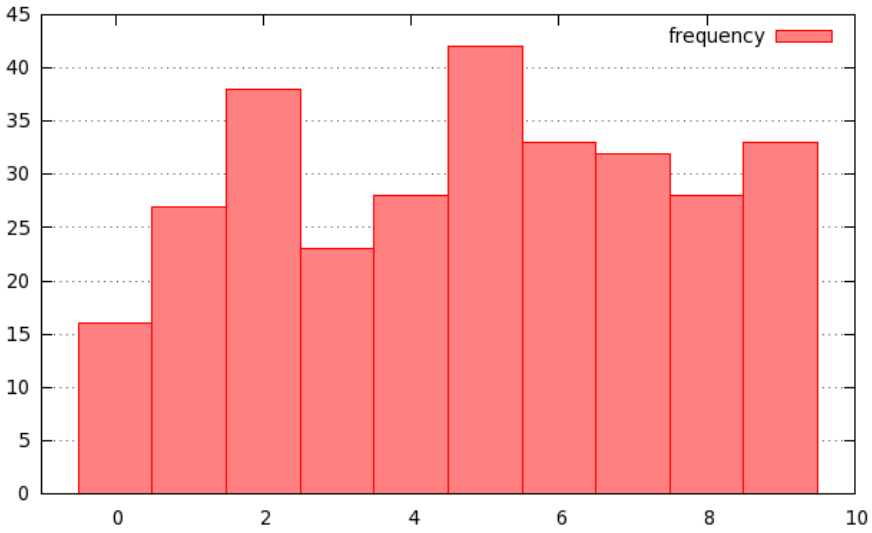
这两种参数都要求有两列数据：X 和 Y。对于同一个 X 值（例如 $X = x_i$），如果有不止一个数据点（$y_{i1}, y_{i2}, \ldots$），那么 frequency 参数就会把 $X = x_i$ 的所有数据点的 Y 值相加，得到一个 $X = x_i$ 处 Y 值的和。如果我们让 $y_{i1} = y_{i2} = \ldots =$常数，那么这个和就可以用来表示数据点出现在 $X = x_i$ 的频率。如果用 cumulative 参数，那么所有 $X < x_i$ 的数据点都会计算在内。

现在我们来看例子：

```
gnuplot> set samples 300
gnuplot> set xrange [1:300]
gnuplot> set format x "%.0f"
gnuplot> set table "random-int.dat"
gnuplot> plot int(10*rand(0))
gnuplot> unset table
```

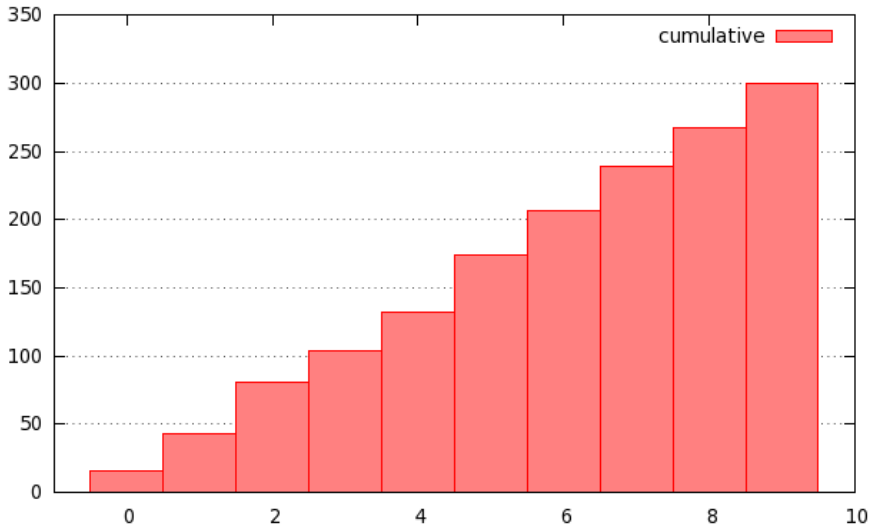
这里我们生成了 300 个 0 到 9 之间的随机整数的序列，并把它存入了名为 “random-int.dat” 的文件。这里的 rand() 是 gnuplot 自带的伪随机数生成函数，能生成 0 到 1 之间的随机数。int() 为取整函数。下面我们来绘图：

```
gnuplot> reset
gnuplot> set boxwidth 1
gnuplot> set style fill solid 0.5
gnuplot> set grid ytics
gnuplot> set xrange [-1:10]
gnuplot> plot 'random-int.dat' u 2:(1) smooth frequency w boxes t "frequency"
```



在 plot 命令里，我们用生成的随机数作为 X，用常数 1 作为 Y，这样 smooth frequency 就给出了 0 到 9 之间各个数字出现的次数。我们再来看 cumulative:

```
gnuplot> set yrange [0:350]
gnuplot> plot 'random-int.dat' u 2:(1) smooth cumulative w boxes t "cumulative"
```

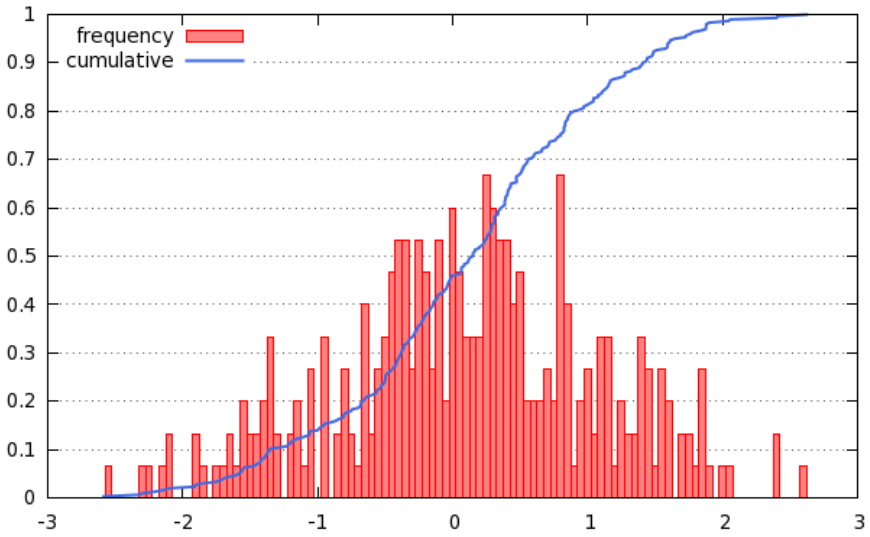


smooth cumulative 把所有 X 小于某一数值的 Y 值累计起来，所以最后达到最大值 300，正好是样本总数。

上面的例子是整数样本，比较容易操作。对于实数（浮点数）样本，我们面临 binning 的问题，这类似于将模拟信号转换为数字信号。下面我们用 gnuplot 源文件中自带的 demo 目录里的数据文件 “random-points” 举个例子。“random-points” 文件包含几组不同分布的随机数，其中第 2 列为正态分布，共 300 个数据。

```
gnuplot> set autoscale
gnuplot> bin(x,s) = s*int(x/s)
gnuplot> set boxwidth 0.05
gnuplot> set key left top
```

```
gnuplot> plot 'random-points' u (bin($2,0.05)):(20/300.) s f w boxes t "
    frequency",\
> '' u 2:(1/300.) s cumul t "cumulative" lw 2 lc rgb "royalblue"
```



注意：我们这里的 plot 命令用了大量缩写。我们先定义了一个函数 bin(x,s)，用于将模拟信号数字化，其中 s 是每个 bin 的宽度（或者叫做样本间隔）。set boxwidth 设定图中每个柱的宽度，我们以前曾经用它设定过相对宽度，这里设定的是绝对宽度。

在 frequency 图中，X 是数字化之后的数据，Y 是常数 20/300.（注意这里的小数点）。分子上的 20 是因为 X 被乘了 0.05，为了保持每个直方柱面积不变（直方图中用面积表示概率），我们把 Y 乘以 20；分母上的 300 是样本总数，用于归一化，这样我们得到的就是分布概率，而不是某区间内数据点的个数。

在 cumulative 图中，X 数据 binning 没有必要，只需要将所有数据累加就可以了，而 Y 是归一化常数。

这样，这两种图的意义就非常清楚了：frequency 给出的是概率密度函数，而 cumulative 给出的是累计分布函数。

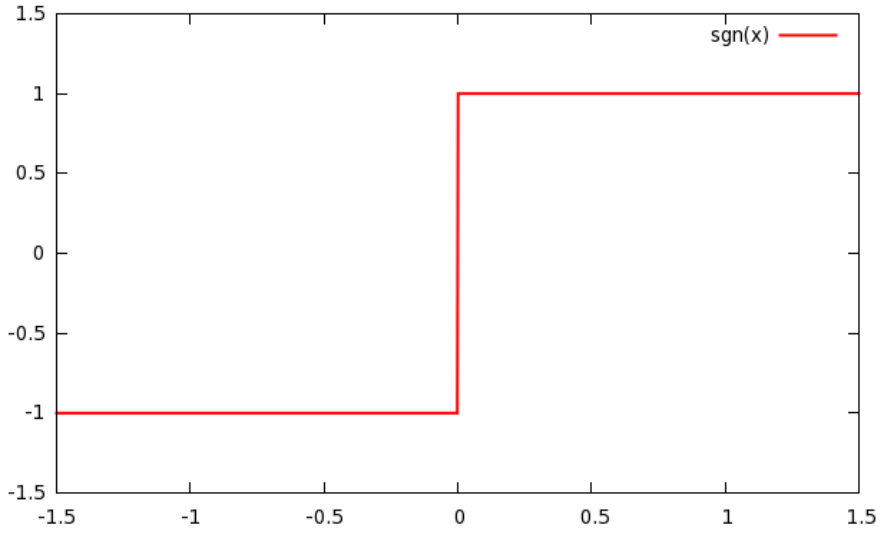
41 三元算符和分段函数

分段函数在若干区间上有不同的表达式，例如符号函数：

$$\mathrm{sgn}(x) = \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ 1 & \text{if } x > 0 \end{cases}$$

gnuplot 预定义函数中包含符号函数 sgn(x)，下面我们试着画一下函数图像：

```
gnuplot> set samples 1000
gnuplot> set xrange [-1.5:1.5]
gnuplot> set yrange [-1.5:1.5]
gnuplot> plot sgn(x) lw 2
```



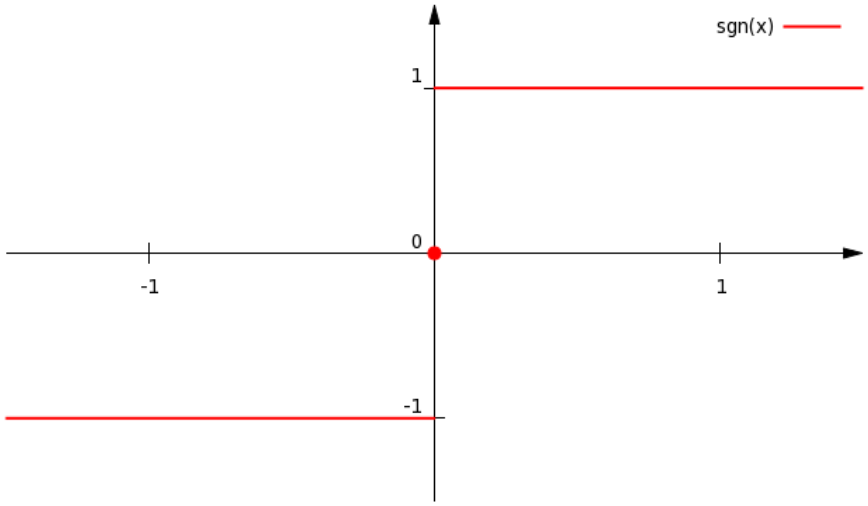
gnuplot 把所有取样点连接了起来，这样函数图像在零点成了连续的，而实际上零点应该是不连续的断点。为了更合理的表现函数性质，我们需要用到三元算符（ternary）。三元算符的形式是：

```
A?B:C
```

A 一般是一个逻辑表达式，如果 A 为真，那么整个表达式的值为 B；如果 A 为假，那么整个表达式的值为 C。这和 C 语言的问号表达式完全相同。

下面我们看如何用三元算符构造分段函数：

```
gnuplot> unset border
gnuplot> set zeroaxis lt -1
gnuplot> set xtics axis -1,2,1
gnuplot> set ytics axis 1 offset 0,0.5
gnuplot> set arrow 1 from 0,0 to 1.5,0 filled
gnuplot> set arrow 2 from 0,0 to 0,1.5 filled
gnuplot> plus(x) = x>0? 1 : 1/0
gnuplot> minus(x) = x<0? -1 : 1/0
gnuplot> set label 1 at 0,0 point pt 7 ps 1.5 lc rgb "red" front
gnuplot> plot plus(x) lw 2 lc rgb "red" title "sgn(x)", minus(x) lw 2 lc rgb "
    red" notitle
```

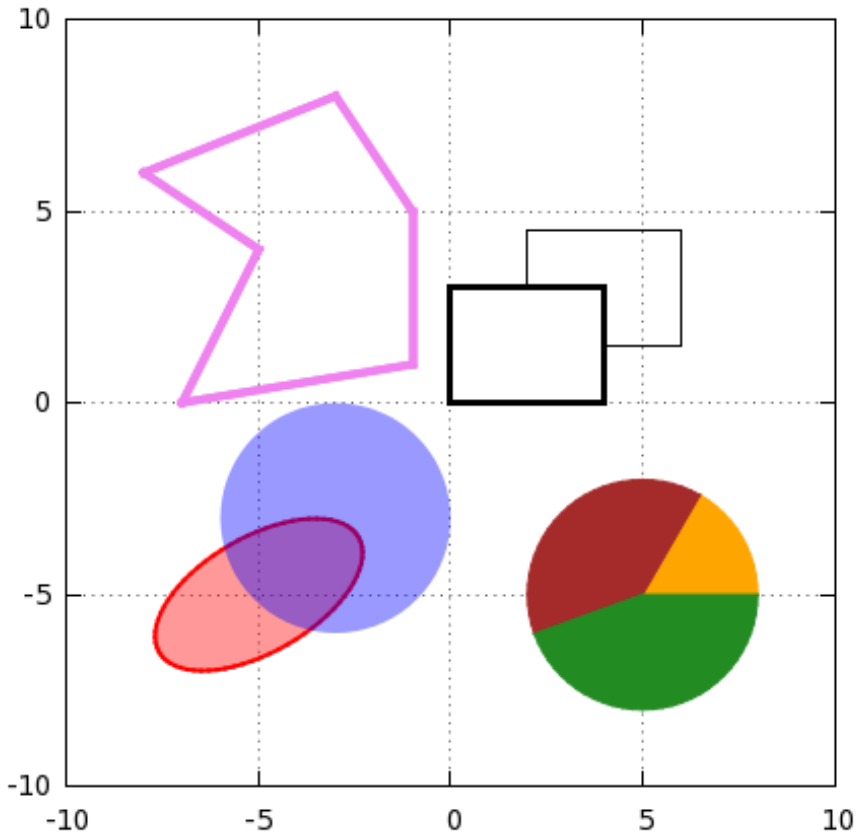


这里我们用 plug(x) 和 minus(x) 分别表示函数的正值和负值部分，用 1/0 表示函数值没有定义。在零点，我们利用一个不含任何字符的 label 命令画一个独立的点。以前我们讲过 label 命令，这里的 label 命令有两个新的参数： point 参数允许在 label 所在坐标画一个点；而 front 参数表示将 label 放在前景，这样它就不会被图像其他部分遮盖，类似于图形处理软件中调整 layer 的顺序。和 front 类似的参数还有 back。

42 几何图形对象

通过设置对象（object），可以让 gnuplot 在图中绘制几何图形。目前 object 支持四种几何图形：rectangle（长方形），ellipse（椭圆形），circle（圆形） 和 polygon（多边形）。下面通过实例说明它们的用法：

```
gnuplot> set grid
gnuplot> set size square
gnuplot> set object 1 rectangle from 0,0 to 4,3 lw 3
gnuplot> set object 2 rectangle at 4,3 size 4,3 behind
gnuplot> set object 3 ellipse at -5,-5 size 6,3 angle 30 lw 2 \
> fillcolor rgb "red" fillstyle transparent solid 0.4
gnuplot> set object 4 circle at -3,-3 size 3 fc rgb "blue" fs transparent solid
    0.4 noborder
gnuplot> set object 5 circle at 5,-5 size 3 arc [0:60] fc rgb "orange" fs solid
gnuplot> set object 6 circle at 5,-5 size 3 arc [60:200] fc rgb "brown" fs
   solid
gnuplot> set object 7 circle at 5,-5 size 3 arc [200:360] fc rgb "forest-green"
    fs solid
gnuplot> set object 8 polygon from -1,1 \
> to -1,5 \
> to -3,8 \
> to -8,6 \
> to -5,4 \
> to -7,0 \
> to -1,1 lw 4 fc rgb "violet"
gnuplot> plot [-10:10][-10:10] 15 notitle
```



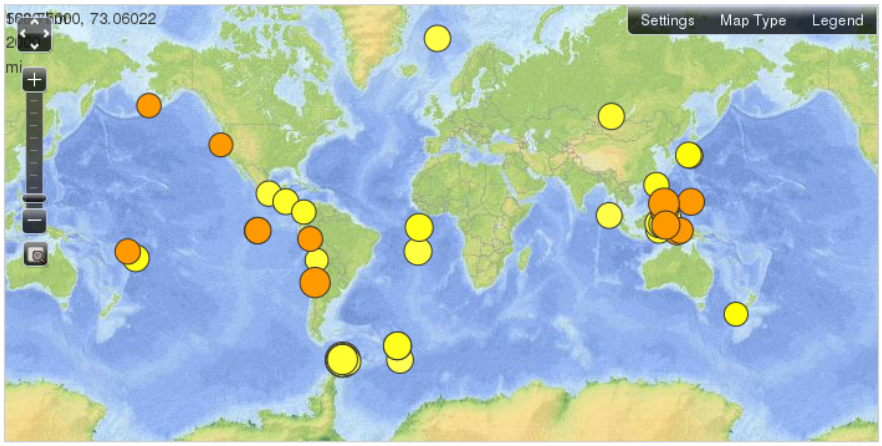
这里共设置了 8 个对象，现在分别说明各个对象的用法：

- 对象 1 和 2 是长方形，既可以通过对角顶点坐标，也可以通过中心坐标和宽高来设置。对象 2 的 behind 参数类似之前讲过的 front 和 back。behind 和 back 的不同在于，back 仅仅是放在曲线和 label 之后，而 behind 是放在所有图像元素之后。
- 对象 3 是椭圆形，参数和长方形类似，多了旋转角度 angle 和填充风格。
- 对象 4 是圆形，通过圆心坐标和半径设置。
- 对象 5、6、7也是圆形，但是多了 arc 参数指定圆弧角度，所以其实画出的是扇形。gnuplot 本身没有“饼图”（pie chart）这种风格，所以利用圆形对象可以绘制简单饼图。
- 对象 8 是多边形，通过指定各顶点坐标来设置，注意多边形必须是闭合的。

设置的各个对象必须通过 plot 命令才能显示出来，而我们又不希望画什么其它图像，所以这里用一种特殊方式，在绘图区域仅显示出各个几何对象。以前我们都是用 set xrange （或 yrange）命令来设置坐标取值范围，其实这个取值范围也可以直接在 plot 命令中给出。这里我们通过在取值范围之外画一条直线的方式，得到了一个仅包含各几何对象的图。

43 地图及圆圈数据图

上次我们讲了几何图形对象，这次我们介绍一种使用几何图形表示数据的方法。下面这幅图来自美国地质调查局网站，显示的是前三天内（至本文写作时）世界各地发生的 4 级以上地震：



现在，我们用 gnuplot 绘制一张类似的地图，用圆圈大小和颜色表示震级，在地图上标注各地震发生的位置。gnuplot 的 demo 目录里有一个 world.dat 文件，包含世界地图的信息，我们就用它来绘制地图。

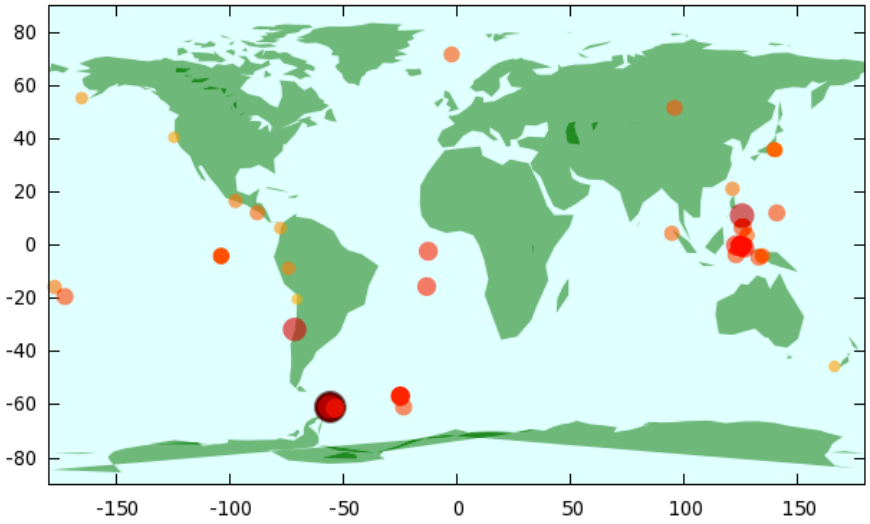
随着我们学习的 gnuplot 命令越来越多，绘制一张图的命令越来越复杂。我们可以把所有命令预先存入一个文件中，随后在 gnuplot 里面调用该文件，这样可以方便我们重复使用这些命令。我们把下面这些命令存入名为 “commands.gnu” 的文件：

```
rgb(r,g,b)=65536*int(r)+256*int(g)+int(b)
red(x) = 2/3.<=x && x<1? 255*(3-3*x) : 255
green(x) = 0<=x && x<1/3.? 255 : x<2/3.? 255*(2-3*x) : 0
blue(x) = 0<=x && x<1/3.? 255*(1-3*x) : 0
circlecolor(x) = rgb(red(($3-2)/5.0),green(($3-2)/5.0),blue(($3-2)/5.0))
unset key
set xrange [-180:180]
set yrange [-90:90]
set style fill transparent solid 0.6 noborder
set object 1 rectangle from graph 0,0 to graph 1,1 fc rgb "light-cyan" behind
plot 'world.dat' with filledcurves lc rgb "forest-green",\
'earthquakes.dat' using 1:2:(($3-3)*2):(circlecolor($3)) with circles lc rgb
    variable
```

随后启动 gnuplot，输入下面的命令：

```
gnuplot> load 'commands.gnu'
```

这里的 load 命令用来调用刚才预存的文件。于是，我们得到下面这幅图：



现在，我们来解释刚才用到的那些命令：

- 1 至 5 行定义了一个从数字到 rgb 颜色的映射，这类似于我们之前“色板设置”里讲过的 rgbformulae，只不过在这里我们使用了自己定义的函数。从这里我们也可以看到 rgbformulae 的工作原理。在函数定义中我们使用了“三元算符和分段函数”讲过的问号表达式，可以作为一种复习。
- 在 set object 命令里我们定义了一个布满整个绘图区域的长方形，并且放置到了 behind 位置，相当于为图像增加了一个背景颜色。
- plot 命令里的前半部分用于绘制地图轮廓，后半部分用于绘制表示地震的圆圈。地震数据来源于美国地质调查局网站，存于名为 “earthquakes.dat” 的文件中。
- 这里用到了一个新的绘图风格：circles。圆圈的大小由 using 参数的第 3 列数据指定，圆圈的颜色由 linecolor 参数设置。然而，这里没有指定具体的颜色，而是用了一个特殊的变量——variable。variable 的值由额外的一列数据（即 using 参数的第 4 列数据）指定，因此颜色会随着不同数据改变。

把我们用 gnuplot 绘制的地图和原图比较一下，地震的位置是一致的。

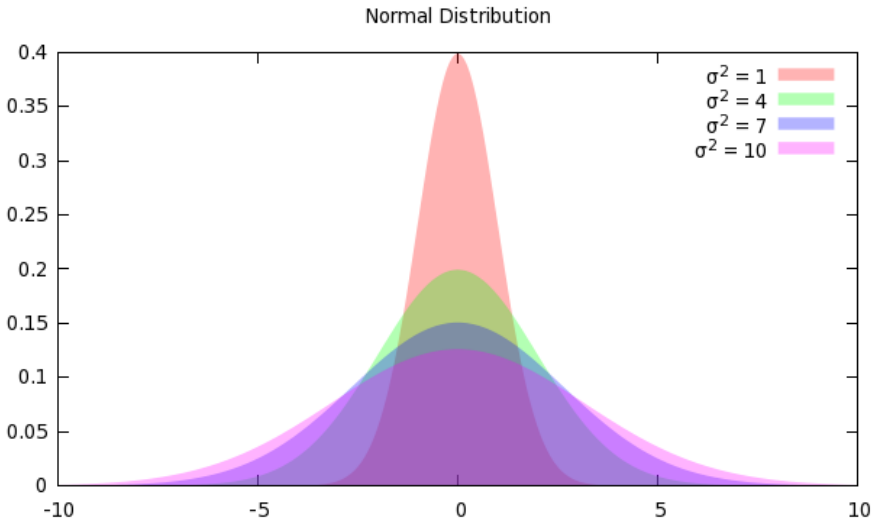
44 for 循环

作为一个具有脚本编程能力的命令行程序，循环语句是必不可少的。gnuplot 具有和 C 语言类似的 for 循环结构，可以应用于 plot, splot, set 和 unset 命令。for 命令的基本结构是：

```
for [i = begin:end:step]
```

这表示 i 的取值从 begin 到 end，步长为 step。如果步长为 1，step 也可以省略。下面举例说明：

```
gnuplot> f(x,s) = exp(-x*x/(2.*s))/(sqrt(2*pi*s))
gnuplot> set term wxt enhanced
gnuplot> set title "Normal Distribution"
gnuplot> set samples 1000
gnuplot> set style fill transparent solid 0.3 noborder
gnuplot> plot for [i=1:10:3] f(x,i) w filledcurves title "σ^2 = ".i
```



plot 命令里的 i 取值分别为 1, 4, 7, 10。值得注意的还有后面的 title 参数，这里的小数点（.）表示将两个字符串连接合为一个。有人可能会想：i 不是整数吗？在 gnuplot 里面，数据类型可能根据环境发生变化，例如这里的整数 i 用在了字符串操作中，就被当作字符串处理了。但是，如果把这个 i 放在小数点前面，这种类型转换就不成立了。我不清楚这种情况的具体成因，但是有一个办法可以解决这个问题，就是在 i 之前再加一个空的字符串，避免 i 成为字符串连接算符（小数点）的第一个元素。

除了上面的形式，for 命令还有一种字符串形式，例如：

```
gnuplot> plot for [filename in "file1.dat file2.dat file3.dat"] filename with
    lines
```

这里就绘制了三个数据文件的图像。

45 动画和 reread 循环

上一讲提到的 for 循环只能用于 plot 等有限的几个命令。如果循环体是一组命令，我们需要用到 reread。reread 的作用就是让 gnuplot 返回到脚本文件最开始的地方，类似于某些编程语言的 goto 命令。为了形成有效的循环（而不是死循环），reread 命令需要配合 if 条件判别式。下面我们利用 reread 命令，举一个生成动画的例子。我们建立两个脚本文件：

1. 文件名 "animate.gnu"：

```
set term gif animate delay 10
set output "animate.gif"
set samples 1000
set xrange [0:4*pi]
unset key
set tics textcolor rgb "orange"
set border lc rgb "orange"
set grid lc rgb "orange"
i=0
load 'looper.gnu'
set output
```

2. 文件名 "looper.gnu"：

```
set object 1 rectangle from screen 0,0 to screen 1,1 fc rgb "gray10"
    behind
plot sin(x+i*pi/20) lw 3 lc rgb "green" notitle
i=i+1
if (i<40) reread
```

为了生成动画，我们使用了一个新的终端：gif 。参数 animate 说明要生成动画，delay 参数指定每两帧图片之间的时间间隔，单位是 1/100 秒。循环体放在一个单独的文件里（looper.gnu），由指标 i 控制循环次数，一共循环执行 40 次。gnuplot里的 if 条件句的用法和 C 语言是相同的，并且也支持 else 和 else if 语句。这里的 if 语句只包含一个命令，如果 if 之后有多个命令需要执行，之间用分号（;）隔开。

现在我们启动 gnuplot，输入下面的命令：

```
gnuplot> load 'animate.gnu'
```

这里是生成的动画： http://image.sciencenet.cn/album/201201/26/024121diddwbfdcbdtmo2i.gif

A 附录：互联网资源

1. not so Frequently Asked Questions
2. Impossible gnuplot graphs
3. Gnuplot tricks blog
4. Gnuplot surprising
5. Gnuplotting

索 引

\, 10
*, 6
**, 6
+, 6
-, 6
/, 6
#, 12

angles, 41
arrow, 33
at, 32
axis, 29, 34

besj0(x), 28, 36
besj1(x), 36
binary, 51
bmargin, 36
border, 34, 52
boxes, 74
boxwidth, 74

cbrange, 55
character, 31
circle, 80
circles, 83
cntrparam, 61
colornames, 63
columnheader, 72
contour, 60
corners2color, 53

ellipse, 80
else, 85
else if, 85
enhanced, 23
eps, 18
epslatex, 26
epstopdf, 27
erf(x), 46
errorbars, 44
errorlines, 45
exit, 5

fc-list, 22
fill, 68
filledcurves, 66
fit, 46
for, 83
format, 39
fsteps, 74

gif, 85
graph, 31
grid, 28, 30, 42

help, 12
hidden3d, 48
histeps, 74
histogram, 70, 76

if, 85
image, 57
increment, 62
int, 76
isosamples, 48

key, 7, 36

label, 31, 80
linecolor, 14
lines, 13
linespoints, 13
linestyle, 14, 36
linetype, 14
linewidth, 14
lmargin, 36
load, 82
logscale, 39

map, 54
multiplot, 65
mxtics, 9

object, 80
offset, 36
output, 19

palette, 56
parametric, 43

pdfcairo, 21
pi, 6
plot, 6
pm3d, 49, 53
pngcairo, 22
points, 13
pointsize, 14
pointtype, 14
polar, 41
polygon, 80
postscript, 18
print, 6, 28

quit, 5

rand, 76
rectangle, 80
replot, 7
reread, 84
reset, 66
rgbalpha, 58
rgbcolor, 31
rgbformulae, 56
rgbimage, 58
rmargin, 36
rotate, 72

samples, 7
screen, 31
set, 7
sgn, 78
size, 40
smooth, 74, 76
splot, 48
steps, 74
style, 36

table, 63
terminal, 5
ternary, 79
textcolor, 32
title, 9, 17
tmargin, 36
trange, 42

unset, 8
userstyles, 62
using, 17

variable, 83
via, 46
view, 49

with, 13
wxt, 5

xerrorbars, 44
xlabel, 9
xrange, 9
xticlabels, 72
xtics, 9
xyerrorbars, 44
xyplane, 48

y2label, 29
yerrorbars, 44
ylabel, 9
yrange, 9
ytics, 11

zeroaxis, 34