

第一部分

GnuPlot 4.6

中文手册 当前修订次数：1

版权

Copyright (C) 1986 - 1993, 1998, 2004, 2007 Thomas Williams, Colin Kelley

Permission to use, copy, and distribute this software and its documentation for any purpose with or without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation.

Permission to modify the software is granted, but not the right to distribute the complete modified source code. Modifications are to be distributed as patches to the released version. Permission to distribute binaries produced by compiling modified sources is granted, provided you

1. distribute the corresponding source modifications from the released version in the form of a patch file along with the binaries,
2. add special version identification to distinguish your version in addition to the base release version number,
3. provide your name and address as the primary contact for the support of your modified version, and
4. retain our contact information in regard to use of the base software.

Permission to distribute the released version of the source code along with corresponding source

modifications

in the form of a patch file is granted with same provisions 2 through 4 for binary distributions.

作者

Original Software:

Thomas Williams,

Colin Kelley.

Gnuplot 2.0 additions:

Russell Lang, Dave Kotz, John Campbell.

Gnuplot 3.0 additions:

Gershon Elber and many others.

Gnuplot 4.0 additions:

See list of contributors at head of this document.

译者：三寸断梁，

前言

本文档针对 4.6 版手册翻译了全文大约 %85 的内容，终端部分从略。本手册是本人空闲时间学习 gnuplot 而翻译，疏漏难免。不保证含义准确，仅供参考。若发现手册中有错误可以报告：gnuplot_cn_manual@126.com，邮件主题必须以"gnuplot:" 开始，否则系统不接收，成功后你可以收到自动回复的邮件。

Gnuplot 是一个小巧的命令行绘图工具，运行于 LINUX，OS/2，MS Windows，VMS 等等其它平台。它的源代码开放。它最初被设计为科学家或学生交互式创建函数图像和数据图表。但是现在它也作为一个绘图引擎用于其它程序，octave 是一个例子。

Gnuplot 支持多种类型的绘图，2D 和 3D。它能利用线段，点，直方图，或者其它元素绘制你需要的图形。同时你可以在图形中添加一些文本。

Gnuplot 支持多种类型的输出类型：交互式屏幕终端（鼠标、键盘），直接输出多种文件格式（png，jpg，LaTeX，pdf，postscript）或者现代打印机。Gnuplot 很容易扩展去支持新的输出类型。最新的支持有基于 wxWidgets 的交互终端，QT。将输出 web 页面可以使用 svg 或者 HTML5 canvas 输出类型。

Gnuplot 是大小写敏感的。所有的命令名和关键字在它无歧义时可以缩写¹。在命令尾部使用分号可以让多条命令出现在同一行。字符串可以用双引号或单引号括主,有少许不同。

如果一条命令太长可以在末尾使用反斜线后在下一行继续(反斜线必须是此行的最后一个字符)：

```
plot sin(x),f2(x),f3(x),\  
f4(x),f5(x);
```

但是如果发生错误，命令解析器无法准确定位到出错的行。在本文中花括号用于表示可选参数，管道符号用于分割一些可选选项。<>用于表示可认为更改的部分。

Gnuplot 在命令行上使用 `gnuplot {OPTIONS} file1 file2. ...;` 启动。对于 X11 系统 可以使用：`gnuplot {X11OPTIONS} {OPTIONS} file1 file2 ...` 具体查阅 X11 的文档。

¹ `plot "data" using 1:2 with lines;` 可以缩写为: `plot "data" u 1:2 w l;`

命令行中的选项可以出现在行的任何地方，文件名将按顺序依次执行。

- 作为文件名，让 gnuplot 从标准输入读取。

新特性

本节介绍 4.4 版本以来主要的附加功能。详见 NEWS 文件。

新语法

这个版本的 gnuplot 新增了迭代指令，和块结构（if else while do）详见后文。简单的迭代指令可以在 plot set 命令中使用。例如：

```
set multiplot layout 2,2
100 term Fourier series
fourier(k, x) = sin(3./2*k)/k * 2./3*cos(k*x)
do for [power = 0:3] {
TERMS = 10**power
set title sprintf("%g term Fourier series",TERMS)
plot 0.5 + sum [k=1:TERMS] fourier(k,x) notitle
}
nset multiplot
```

本地化的 linetypes 用户设置

你可以设置默认的 linetype，详见 set linetype。这通常在启动文件中设置。

新的 plot styles

查看 plot styles：boxplot，circles，ellipses 的文档。

翻转坐标轴

坐标轴现在可以不使用 x y 命名。详情查看 set polar 和 set rrange

新的平滑算法

新的算法支持 2d 3d 绘制。smooth kdensity 或者 smooth cumulative 同 plot 指令同时使用。

新的时间和日期处理

Gnuplot 现在使用毫秒（千分之一秒）精度。时间数据必须被格式化成规定格式。内建函数 `time()` 返回当前系统时间。例如输出时间：

```
print strftime("%H:%M:%.3S %d-%b-%Y",time(0.0))  
18:15:04.253 16-Apr-2011
```

数据统计汇总

新命令 `stats` 从文件读取数据，使用 `plot` 相同的语法。并且输出统计汇总，包括最大值，最小值，众数，平均值，标准差，关联性。等等。

反向兼容

Gnuplot 4.0 摒弃了旧版本中的部分语法。但是仍然对它们提供了支持。需要在编译时提供支持：`./configure --enable-backwards-compatibility`。

Deprecated:

```
set title "Old" 0,-1  
set data linespoints  
plot 1 2 4  
# horizontal line at y=1
```

New:

```
TITLE = "New"  
set title TITLE offset char 0, char -1  
set style data linespoints  
plot 1 linetype 2 pointtype 4
```

批量处理和交互操作

Gnuplot 既可以从文件读取命令执行，也可以交互运行。使用 `-e "command"` 将执行一个 gnuplot 命令。

使用 `-persist` 直接从命令行读取程序。退出后窗体后台保留。

```
gnuplot -persist -e "set title ' Sine curve' ; plot sin(x)"
```

设置变量影响脚本执行。

```
gnuplot -e "a=2; s=' file.png' " input.gpl
```

画布尺寸

以前版本 gnuplot，对于某些终端类型使用 `set size` 命令控制画布大小，而其它终端用其它方法。4.2 版本以来 `set size` 命令被放弃。4.4 版本几乎所有的终端类型使用同样的语法。

```
set term <terminal type> size <XX>, <YY>
```

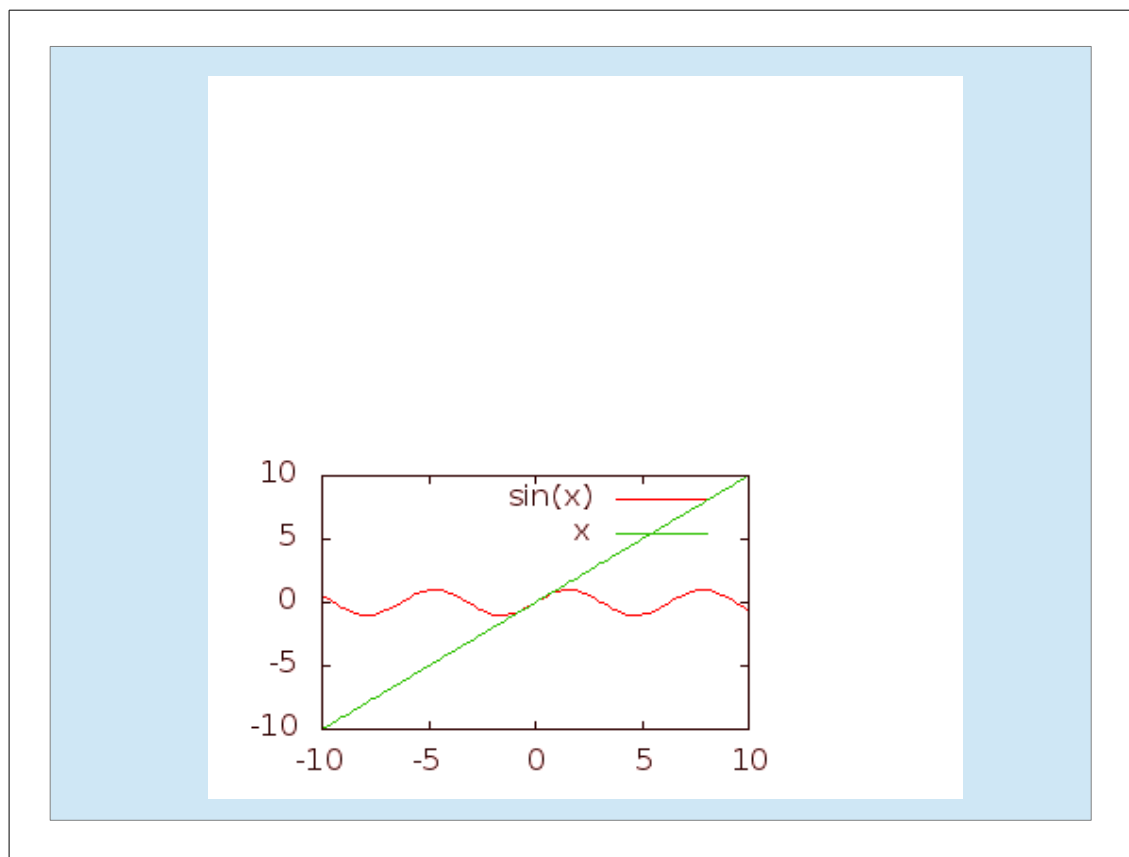
默认情况下，输出将使用整个画布。

`set size <XX>,<YY>` 缩放 plot 相对于画布的大小。值小于 1 表示 plot 不使用整个画布。值大于 1 表示 plot 的某一部分将会被显示在画布上。在某些终端类型下值大于 1 会出错。

这个规则的主要例外是，PostScript 驱动。

```
set size 0.8, 0.5  
set term png size 400, 400  
set output "figure.png"  
plot sin(x), x
```

这些指令用于绘制一个 400*400 大小的 png 图片，绘制区域横向使用 80% 纵向使用 50%（下图蓝色边框是译者加入的）：



注释

脚本中的注释 #开始一直到行尾。

数据文件中的注释由 set datafile commentschars 设定，默认为#。

坐标

set arrow, set key, set label 和 set object 命令允许使用坐标作为参数，坐标的表示方法为：

`{<system>} <x>, {<system>} <y> {,{<system>} <z>}`

<system> 可以为 first, second, graph, screen, 或者 character.

First 设置 xyz 值到第一坐标系¹，second 设置 xyz 到第二坐标系，graph 设置相对与图像区域的比例，0,0 代表左底点,1,1 右上点。screen 代表坐标相对与整个画布²，0,0 代表画布左底点,1,1 画布右上点。character 关键字，坐标位置决定于字符的宽或高，因此它和当前字体相关。参考：page 7

如果 x 没有指定坐标系，默认使用 first，如果 y 没指定，默认依赖于 x。

某些情况下需要使用相对坐标（例如 set arrow 的第二个参数）。如果给定坐标处于对数坐标系，那么值被解释为比例因子。例如：

```
set logscale x
set arrow from 100,5 rto 10,2
```

箭头从 100,5 指向 1000,7

如果某个轴的数据是日期，那么必须使用满足 set timefmt 格式的字符串。

字符串数据

数据文件现在可以包括字符串数据。字符串可以包括任意可显示字符。

1.000 2.000 "NUM ONE" 2.00

2,000 1.000 "N. TWO" 3.00

此行数据含有 4 列。

2D 或者 3D plot 指令可以将文本字段自动定位。

```
plot 'datafile' using 1:2:3 with labels
```

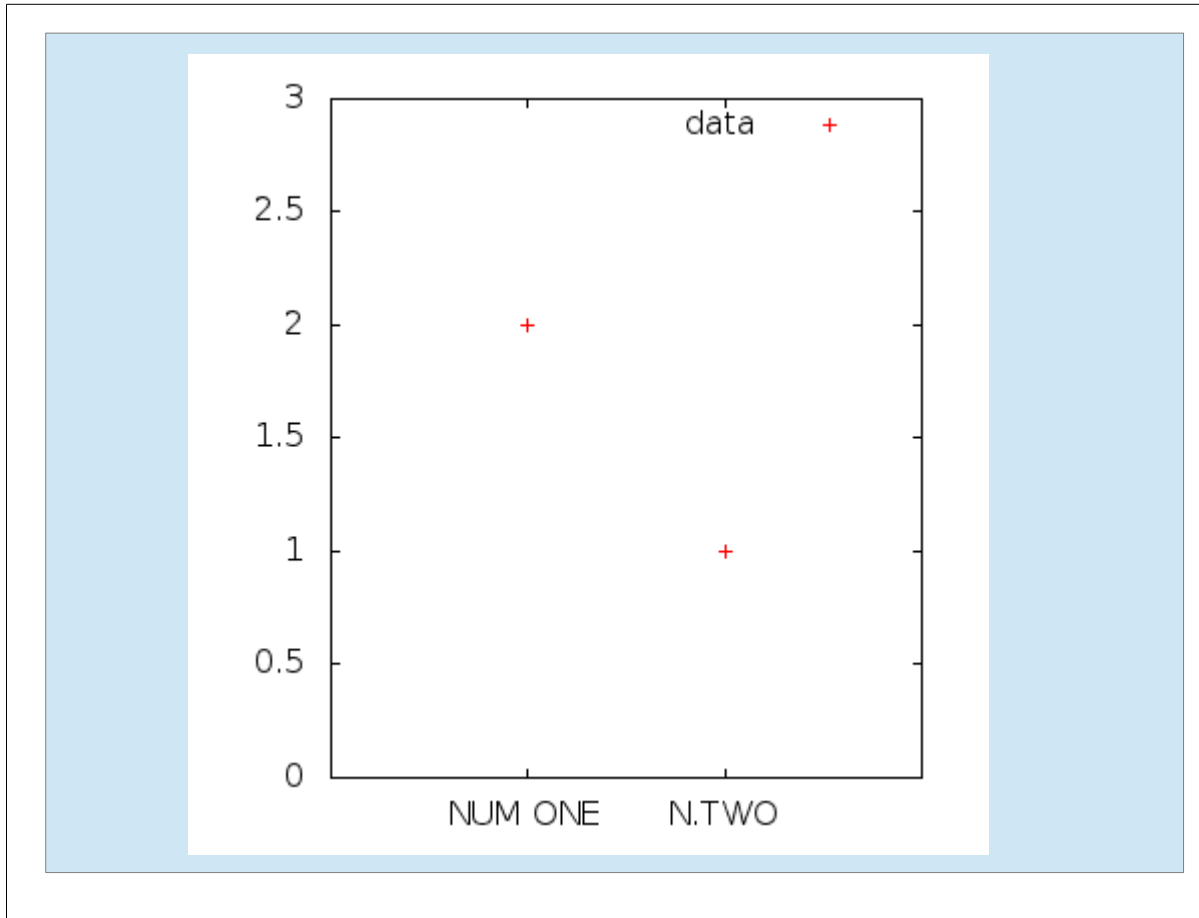
1 gnuplot 有两个坐标系统参见 24

2 画布和图像区域，前者范围大于图像区域。系统在图像区域之外的画布区域为坐标轴注明刻度标签。

这些命令将从文件中读取数值作为坐标，将字符串写在指定坐标处。

读取字符串并将它写在横轴标点,可用于直方图的命名。

```
reset
set xtics
set term png size 400,400
set output 'f.png';
plot [0:3][0:3] "data" using 1:2:xticlabels(3) title "data"
```



注：图中蓝色区域为译者加入，为了让读者看到图像边界。坐标轴以内的区域为 graph 区域（图像区域），之外的白色区域为 screen 区域（画布区域）。

增强文本模式

与 tex 兼容的表达方法用于输入数学公式。花括号用于字符分组。例如：

a^x+2 a^x+2

a^{x+2} a^{x+2}

a^{10} a^{10} 注意

这个 10 是作为两个元素处理的。

$a^{\{10\}}$ a^{10}

set termoption enhanced 开启增强模式

set label 'x 2' noenhanced 关闭增强模式

想指定字体或者字体大小请使用完整的格式：`{/[fontname][=fontsize | *fontscale] text}` 例如：`{/Symbol=20 G}` 表示 20pt 的字符 G，`{/*0.75 K}` 字符 K，尺寸为当前尺寸的 3/4。强调一点/符号必须是花括号内第一个字符，也就算说{和/必须紧挨着。这对括号对上标和下标非常有用，但是对于单词的重音符号就不好用了。对于这种字符最好使用特殊文本编码 iso 8859 1 or utf8。这些字符集里包含了大量有重音和附加其它标记的字符。产生空白可以使用 `&{text}` 它将产生与 text 等宽的空白。
' abc&{def}ghi' 生成 ' abc ghi'。

~符号用于让下一个字符或者元素在同一水平位置输出，换句话说让他们重合，用于产生标注字符。

"~{abc}{1+++}" $\overset{+++}{abc}$ 加号提升 1 个单位。

"~{abc}{.7 ooo}" $\overset{ooo}{abc}$ 小写 o 提升 0.7 个单位。这个不美观。

"~a{1/* .5 o}" $\overset{\circ}{a}$ 小写 o 提升 1 个单位，尺寸减半。

"~a{0/*1 o}" $\text{\textcircled{a}}$ 小写 o 与小写 a 完全重合。

"~a{0/*2 o}" $\text{\textcircled{a}}$ 小写 o 放大一倍与 a 完全重合。

使用这个功能你可以组合出你想要的特殊符号。

你可以使用 `\XXX` 访问字符，类似 c 语言中的表示方法，它直接表示字符编码值。

`{/Symbol \245}` 一个无穷的符号。但是这个方法在多字节编码的文档中工作不好，例如 utf8。这不会造成什么限制，在 utf8 中你就直接输入这个字符就可以了。

Control	Example	Result
<code>^</code>	<code>a^x</code>	a^x
<code>-</code>	<code>a_x</code>	a_x
<code>@</code>	<code>a@^b_{cd}</code>	a^b_{cd}
<code>&</code>	<code>d&{space}b</code>	$d_{\text{space}}b$
<code>~</code>	<code>~a{.8-}</code>	\tilde{a}

使用\ 去转义一个元字符。

环境变量

Gnuplot 可以理解一部分 shell 变量。但对 shell 变量不存在什么依赖，有些 shell 变量对你应该有用。如果 GNUTERM 被定义，它的值将成为 gnuplot 启动时的默认终端。但是如果 ./gnuplot 或者其它配置文件存在，文件中的设置将覆盖这个值。

GNUHELP 变量定义 HELP 文件 (gnuplot.gih) 的路径。

省略三个变量的翻译：GNUPLOT_PS_DIR GNUPLOT_LIB FIT SCRIPT

表达式

许多编程语言都支持数学表达式。运算符处理规则与 C 编程语言一致。空白字符 (空格, tab) 在表达式内忽略。

复数用{实数,虚数}表示法。{0,1}表示 i 也就算虚数单位 $\sqrt{-1}$ 。{3,2}==3+2i .花括号不能省略。gnuplot 使用整型和浮点数，类似于 C 和 FORTRAN 语言。整数 1 -10，浮点数 1.0 -10.0 1e-3。需要注意的是 gnuplot 也有两种除法，整数除法和小数除法：5/2==2；5.0/2==2.5。对于负整数除以正整数，例如-5/2，结果可能是-3 或者-2，这跟编译器相关。

除零，1/0 将设置 undefined 标志。可以使用内建的 NaN 来表示这个值。

对于虚数{x,y}，x y 一定会被转换成浮点数，{3,2}=={3.0,2.0}。

单个小数点用于连接字符串 "AAA" . "BBB" eq "AAABBB",小数点是字符串连接运算，eq 是字符串比较运算。当字符串包含一个数字，它用于数学表达式时能自动转化成浮点数或整型。"3"+"7"==10；6.78=="6.78"。数字用于字符串运算也能自动转换成字符串，"file" . 4 eq "file4"；字符串索引，"string"[3:4]=="ri" "ABCDEF"[4:]*=="DEF"，字符串索引使用了内建的 substr(str,begin,end)函数。

函数

gnuplot 的函数和 unix 数学库相匹配。所有的函数都接受 整数 浮点数 复数，无需手动指明类型。

有些函数接受或者返回 角度(sin(x) asin(x))角度单位可以为弧度或者度(弧度 2π ==360 度)。

函数

参数

abs(x)	任何	实数绝对值
abs(x)	复数	{x,y}到原点的距离。
acos(x)	任何	cos x 反函数
acosh(x)	任何	cosh x 反函数 返回弧度。
arg(x)	复数	x 的相
asin(x)	任何	sin x 反函数
asinh(x)	任何	sinh x 反函数, 返回弧度。
atan(x)	任何	tan x 反函数
atan2(y,x)	实数	tan (y/x)反函数
atanh(x)	任何	tanh x 反函数返回弧度
EllipticK(k)	$k \in (-1:1)$	K(k) 第一类完全椭圆函数
EllipticE(k)	$k \in [-1:1]$	E(k) 第二类完全椭圆函数
EllipticPi(n,k)	real n<1, real $k \in (-1:1)$	$\Pi(n, k)$ 第三类完全椭圆函数
besj0(x)	实数	j0 Bessel 函数, 弧度。
besj1(x)	实数	j1 Bessel 函数, 弧度。
besy0(x)	实数	y0 Bessel 函数, 弧度。
besy1(x)	实数	y1 Bessel 函数, 弧度。
ceil(x)	任何	[x],不小于 x 的最小整数。
cos(x)	任何	余弦
cosh(x)	任何	双曲线余弦函数
erf(x)	任何	erf(real(x)), error function of real(x)
erfc(x)	任何	erfc(real(x)), 1.0 - error function of real(x)
exp(x)	任何	e^x
floor(x)	任何	不大于 x 的最大整数。
gamma(x)	任何	gamma 函数
ibeta(p,q,x)	任何	
inverf(x)	任何	
igamma(a,x)		

imag(x) 复数 返回复数的虚数部分。

invnorm(x)

int(x) 任何 截断实数的小数部分

lambertw(x) Lambert W

lgamma(x)

log(x)

log10(x)

norm(x) 任何 Gaussian 函数的普通版本

rand(x) 产生随机数

real(x) 任何 转换成实数。

sgn(x) 任何 负数返回-1，正数返回 1，0 返回 0。虚数忽略。

gprintf("format",x) gnuplot 的格式化输出函数。

sprintf("format",x,...) C 语言格式化输出函数。

strlen("string") 字符串长度

strstr("string","key") key 在 string 中的索引。

substr("string",beg,end) 返回子字符串，"string"[beg:end]; substr("abcdef",3,5)=="cde"

strftime("timeformat",t) t 为时间秒数得到。格式化时间输出。

strptime("timeformat","s") s 为表示时间的字符串。例如："2012-07-30 22:59"

system("command") 运行系统函数，并且返回标准输出。

word("string",n) 返回 string 中第 n 个单词

words("string") 返回 string 中有多少单词。

column(x) 返回数据库中第 x 列

defined(X) 返回变量 X 是否定义。定义返回 1 否则返回 0。

exists("X") 返回名为 X 变量是否存在，存在返回 1 否则返回 0

stringcolumn(x) 列 x 作为字符串返回。

timecolumn(x) 数据库中第 x 列作为时间返回。

tm_hour(x) x 时间数据的小时部分。

tm_mday(x)	返回月份的几号。
tm_min(x)	分钟
tm_mon(x)	月份
tm_sec(x)	秒
tm_wday(x)	星期几
tm_yday(x)	当年的第几天，例如 x 是第 265 天返回 265
tm_year(x)	返回年份
valid(x)	判断第 x 列是否有效。

椭圆函数

$\text{EllipticK}(k) = (1 - (k \cdot \sin(p))^2)^{-0.5}$. k 介于 $0, \pi/2$

$\text{EllipticE}(k) = (1 - (k \cdot \sin(p))^2)^{0.5}$ k 介于 $0, \pi/2$

$\text{EllipticPi}(n, k) = (1 - (k \cdot \sin(p))^2)^{-0.5} / (1 - n \cdot \sin(p)^2)$ $n < 1, -1 < k < 1$

$\text{EllipticPi}(0, k) == \text{EllipticK}(k)$

产生随机数

rand(0) 产生[0,1]的伪随机数，利用两个内建的 32bit 种子。

rand(-1) 设置随机种子为标准值

rand(x) $x > 0$ 根据 x 产生种子。

rand({x,y}) $x > 0$ 种子 1=x 种子 2=y。

运算符

一元运算符

- -a 取反，-1 就是负一，

+ +a 正数，无须运算。

~ ~a 二进制补数。也就是按位取反。a 必须整型

! !a 逻辑否定运算。a 必须整型

! a! 阶乘 a 必须整型，但返回浮点数，这样可以有更大的返回值取值范围。

\$ \$3 引用参数，或者 using 语句中的列。参数必须整型

前缀运算符的运算级别和结合性与 C Fortran 一致。 $-2^{**}2 = -4$, but $(-2)^{**}2 = 4$.

二元运算符

下表运算优先级降序排列。

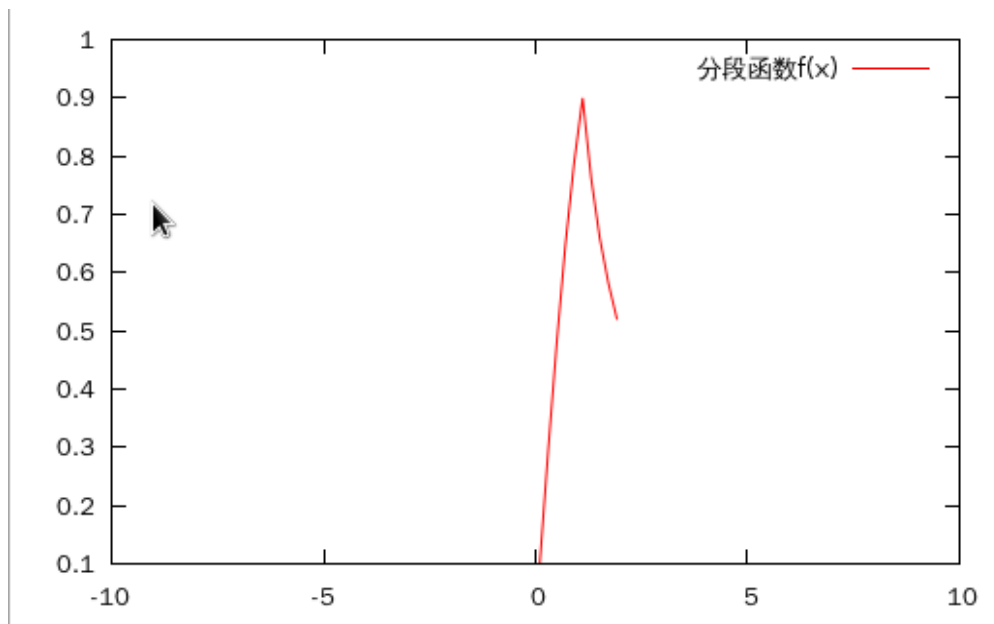
数学运算		位运算	
a**b	乘方运算	a&b	按位与，and
a*b	乘法	a^b	按位异或
a/b	除法	a b	按位或
a%b	取模，也就是余数	逻辑运算	
a+b	加法	a&&b	条件且
a-b	减法	a b	条件或
比较运算		a = b	赋值运算
a==b	等于	(a,b)	逗号表达式，类似 C 语言
a!=b	不等于	字符串运算	
a<b	小于	A.B	字符串连接
a<=b	小于等于	A eq B	字符串等于
a>b	大于	A ne B	字符串不等
a>=b	大于等于		

与 C 语言相同，a&&b 运算当 a 不成立，b 将不再求值。a||b a 成立，b 不再求值。

三元运算

$a ? b : c$ 与 C 语言一样 当 条件 a 成立返回 b 的值，不成立返回 c 的值。定义一个分段函数：

```
f(x) = 0 <= x && x < 1 ? sin(x) : 1 <= x && x < 2 ? 1/x : 1/0  
plot f(x) title '分段函数 f(x)'
```



gnuplot 安静的忽略未定义的值，分支最后的 $1/0$ 将产生无法绘制的点，没有提示。应用了 line 样式, $f(x)$ 作为连续函数绘制, 自动连接不连接部分. 创建非连续分段函数

绘制数据, 纵坐标为 $(\$2 + \$3)/2$, 横坐标为 $\$1$. 当 $\$4$ 不小于 0.

```
plot 'file' using 1: ( $4 < 0 ? 1/0 : ($2+$3)/2 ).
```

Gnuplot 内建变量

Gnuplot 管理着一些只读变量, 他们反应着程序当前状态。变量都有前缀 GPVAL_ 例如:

GPVAL_TERM, GPVAL_X_MIN, GPVAL_X_MAX, GPVAL_Y_MIN . 使用内建命令

show variables all 显示所有变量.

如果最近一个命令执行出错, GPVAL_ERRNO 被设置为非零值。 GPVAL_ERRMSG 保存对应的出错信息。命令 reset errors 清除这些信息为空。

在支持鼠标的交互终端下支持 mouse variables 。这些变量以 MOUSE _ 开头。详情见后文。

用户定义变量和函数

用户函数最多可使用 12 个参数。他们可用于任何地方，包括 plot 命令。合法的变量名就和 C 或者其它高级语言一样，由字母数字 下划线组成，但是首字母不能是数字。语法：

函数名(<dummy1> {,<dummy2>} ... {,<dummy12>})=表达式

表达式由 dummy1 到 12 和自他运算符，函数组合而成。

```
w = 2
q = floor(tan(pi/2 - 0.1))
f(x) = sin(w*x)
sinc(x) = sin(pi*x)/(pi*x)
delta(t) = (t == 0)
ramp(t) = (t > 0) ? t : 0
min(a,b) = (a < b) ? a : b
comb(n,k) = n!/(k!*(n-k)!)
len3d(x,y,z) = sqrt(x*x+y*y+z*z)
plot f(x) = sin(x*a), a = 0.2, f(x), a = 0.4, f(x)
```

```
file = "mydata.inp"
file(n) = sprintf("run_%d.dat",n)
```

最后两个例子就是用户定义字符串，和用户定义字符串函数。圆周率 pi (3.14159...) and NaN (IEEE "Not a Number" 自动定义。你可以修改他们：

```
NaN = GPVAL_NaN
pi = GPVAL_pi
```

字体

Cairo (pdfcairo, pngcairo, wxt 终端)

这一节正还在组织。终端寻找和访问字体通过外部程序 fontconfig 。可以在终端后加上字体选项：

```
set term pdfcairo font "sans,12"
set term pdfcairo font "Times,12"
set term pdfcairo font "Times-New-Roman,12"
```


Gd (png, gif, jpeg terminals)

png , gif jpeg 终端字体的处理使用外部库 : libgd。有五种字体 libgd 内部提供 : tiny (5x8 pixels), small (6x12 pixels), medium, (7x13 Bold), large (8x16) or giant (9x15 pixels) 。这些字体不能缩放和旋转。使用后缀关键字使用他们 :

```
set term png tiny
```

当然可以使用外部字体 :

```
set term png font '/usr/share/fonts/wqy-zenhei/wqy-zenhei.ttc'
```

大部分系统的 libgd 允许访问 Adobe Type 1 字体 (*.pfa) 和 TrueType 字体 (*.ttf) , 你必须指定字体文件名, 而不是文件内的字体名。格式 <face> {,<pointsize>} , 表明字体信息, face 为字体名, 或绝对路径, pointsize 为字体尺寸。当只给出文件名 face 而没有指定路径时, gnuplot 将搜索 GDFONTPATH 下所有路径下文件为 face.ttf face.pfa 的字体。

```
set term png font "arial,11"
```

TTF 和 pfa 字体都是可缩放和旋转的。如果 set term 内没有指定字体, GNUPLOT_DEFAULT_GDFONT 环境变量中保存着默认字体名。

新版本的 gnuplot 支持中文, 但是你的 libgd 并不一定能处理中文。所以最好将输出终端设置为 eps 格式, 并且设置中文字体。

Postscript encapsulated postscript *.eps

PostScript 字体可以用于打印机和浏览程序。即使你的系统里没有安装所有需要的字体 Gnuplot 也可以生成 PostScript 文件, 和 eps 文件。Gnuplot 将在 ps 文件里保存字体名, 让打印机或浏览程序找到最接近的字体。

所有 Postscript 打印机都包含标准 Adobe 字体 Times-Roman, Helvetica, Courier, and Symbol 。当然打印机也支持其它字体, 这取决于你的系统和打印机的配置。Gnuplot 不关心这些信息。因为它不需要知道。生成的 *.ps or *.eps 文件包含一些信息去表明需要那些字体。

```
set term postscript eps font "Times-Roman,12"
```

将产生所有打印机都可用的文件。因为 Times-Roman 字体大家都支持。

```
set term postscript eps font "Garamond-Premier-Pro-Italic"
```

这条命令产生一个 eps 文件。但由于它使用了一个特殊字体, 所以只有一部分打印机或浏览器能正确处理。大部分情况这个特殊字体被内置字体取代。当然可以将一个字体嵌入到 eps 文件中, 这样所有打印机都能正确处理了。这需要你的系统有一个合适的字体描述文件。要注意一点: 一部分字体嵌入时需要一个证书。

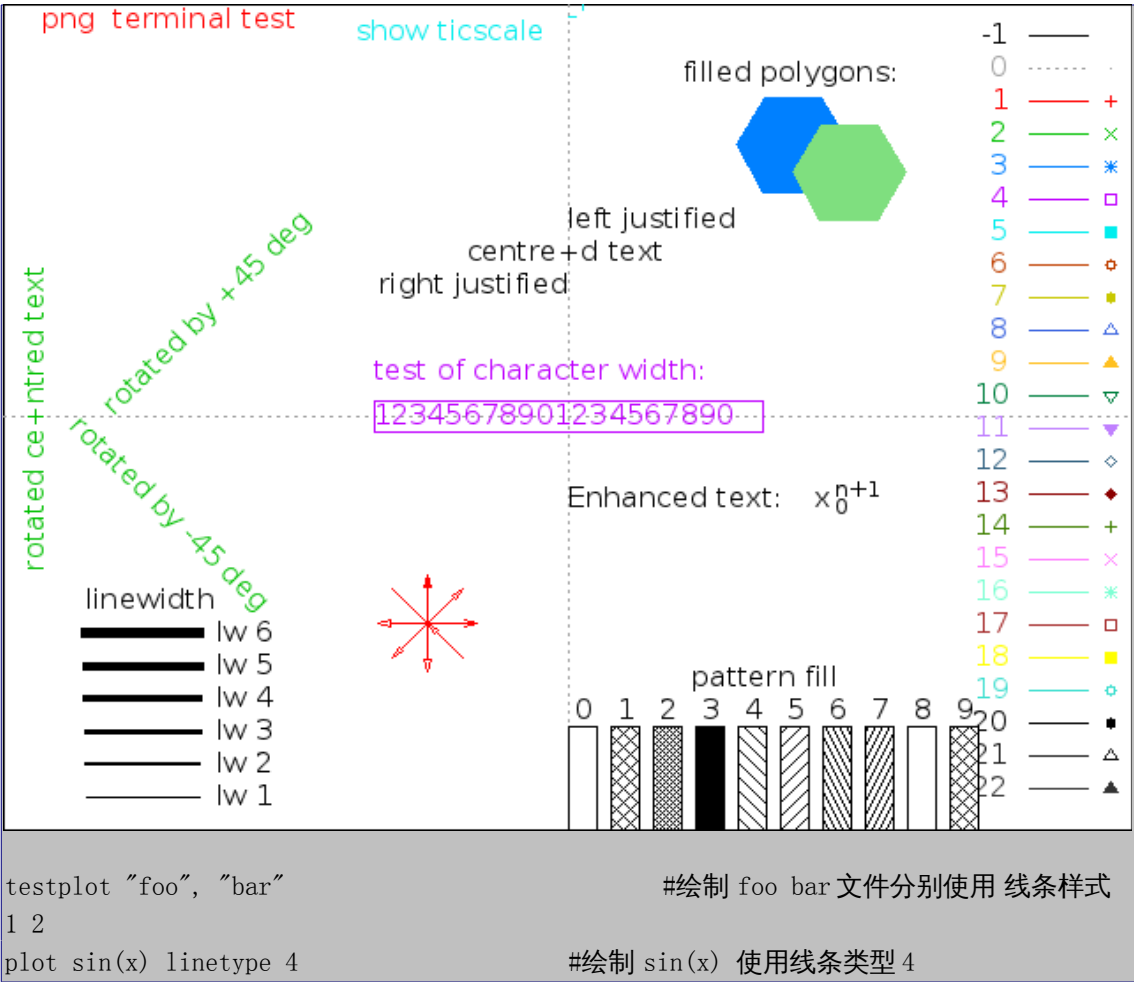
特殊词汇

这章用来解释一些特殊词汇。

page screen canvas , 页面 屏幕 画布 , 表示 gnuplot 能定位的整个区域。在桌面上是整个窗体 , 在绘制器上就是一页。svga 模式下是整个显示器。一个屏幕包含若干 图 , 一个图由横坐标和纵坐标定义 (他们不一定要出现在图上) , 图的边沿是空白 , 用来写一些文字 , 比如图的名字和标注坐标轴。一副图包含一个图象 , 由横坐标和纵坐标定义 (坐标轴不一定要出现在图象上) 。一个图象包含一些线条或者各种形式的点。他们由 plot 语句根据函数或者数据文件绘制。图象中的线条或者点有自己的名字。他们可以用同一个样式 绘制 , 样式常常作为关键字指定。

线条样式, 色彩, 样式

每个 gnuplot 终端类型都提供了一系列线条样式。它们在颜色粗细和点的模式上不一样。对于某个终端类型使用 test 命令观察默认样式。test



```

plot sin(x) lt -1 #绘制 sin(x) 使用线条类型-1，默认样式。lt 为 linetype 的简写。
plot sin(x) lt rgb "violet"          #绘制 sin(x) 使用指定颜色, violet 为内建色彩名
plot sin(x) lt rgb "#FF00FF"        #绘制 sin(x) 使用指定颜色
plot sin(x) lt palette cb -45        #使用调色板计算颜色, 详情见后文.
plot sin(x) lt palette frac 0.3      #使用调色板计算颜色, 详情见后文.

```

使用 `show colormnames` 命令查看预定义色彩名.

使用 `linecolor` 属性覆盖色彩.

```

set term postscript dashed color
plot 'foo' lt 3, 'baz' lt 3 linecolor 1, 'bar' lt 3 lc rgb 'gold'
#lc 为 linecolor 简写

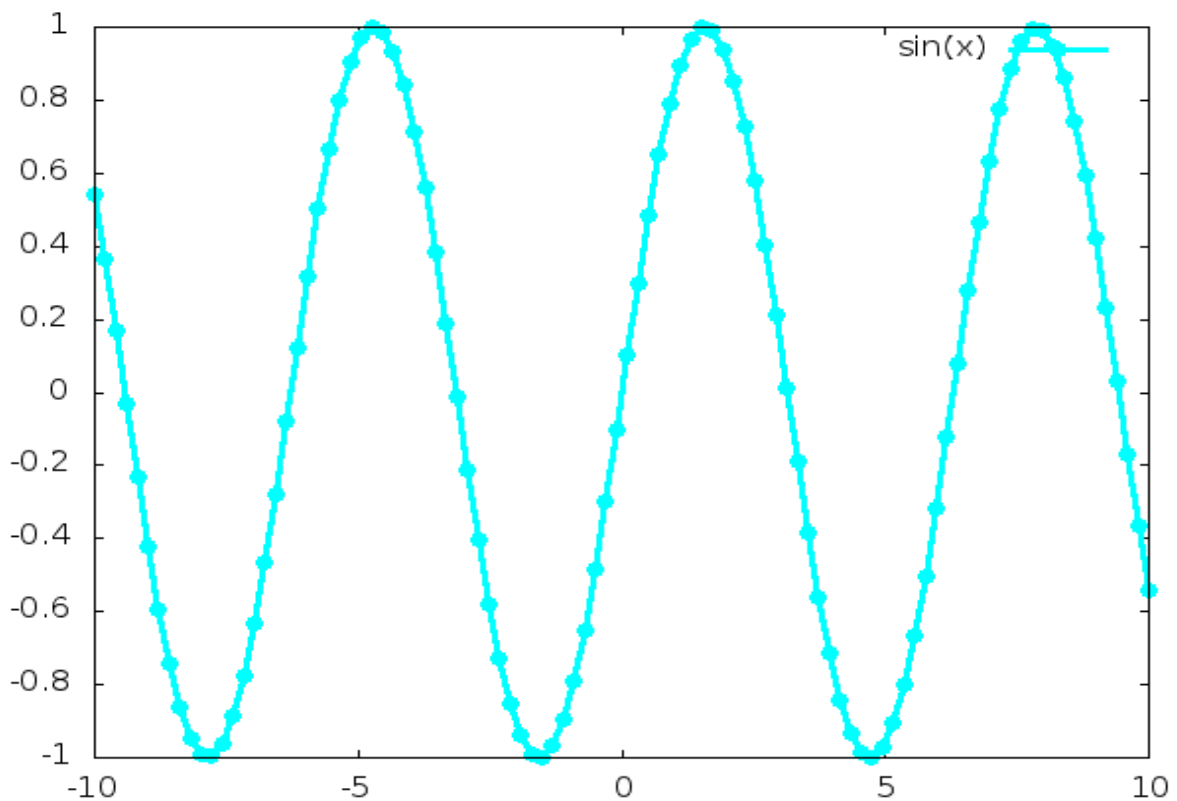
```

线条样式还有其它附加属性例如 `linewidth`，线条宽度。你可以使用 `set style line` 命令设置自定义线条样式。

```

set style line 5 lt rgb "cyan" lw 3 pt 6 # 设置样式 5 的属性
plot sin(x) with linespoints ls 5        #使用线条样式 5.

```



颜色指定

许多命令可以为线条样式指定颜色。非终端依赖的色彩选项允许使用 RGB 色彩和 pm3d 调色板.

```

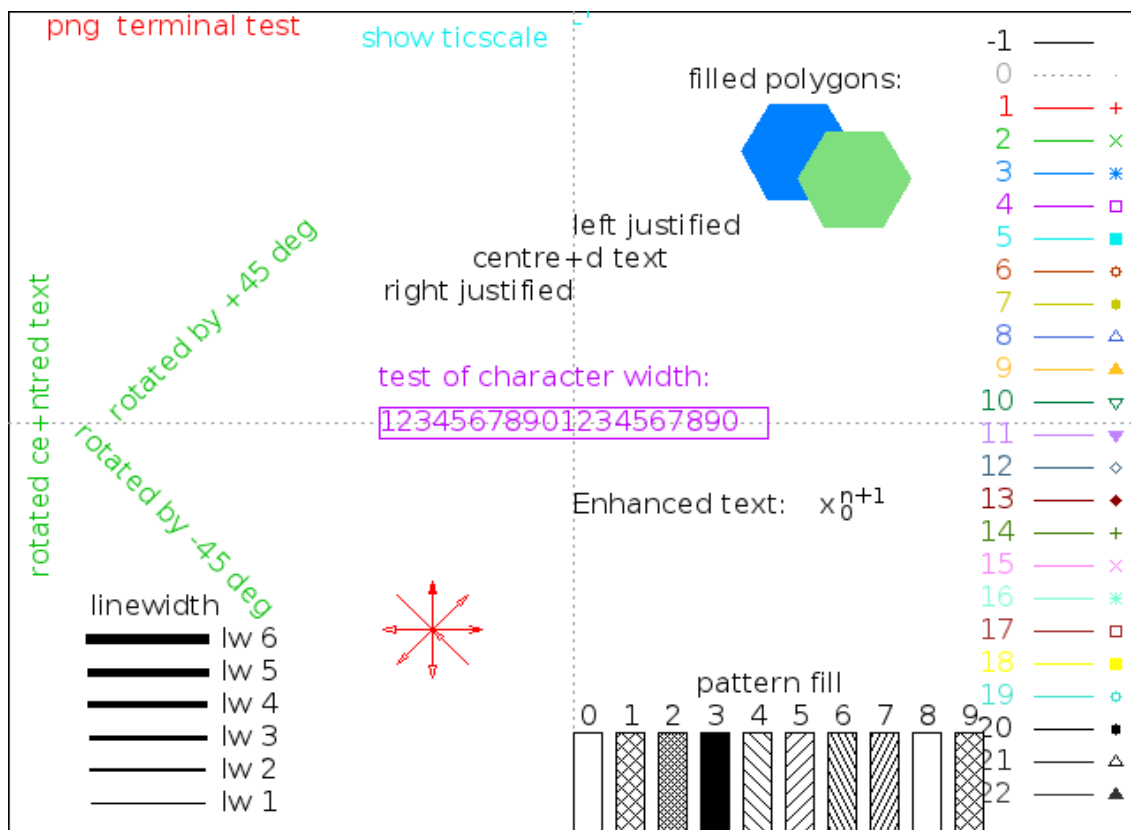
... {linecolor | lc} {<colourspec> | <n>}
... {textcolor | tc} {<colourspec> | {linetype|lt} <n>}

```

`colourspec` 可以用下面的格式指出：

`rgbcolor "colorname"`
`rgbcolor "#RRGGBB"`
`rgbcolor variable` #色彩从输入文件中读取。
`palette frac <val>` #val 介于 0 1 之间
`palette cb <value>` #value 依赖 cbrange
`palette z`
`variable` #色彩索引从文件读取。

<n> 是一个线条样式的编号。参看 test 的输出，右边的 1 2 3 4。



#RRGGBB 是一个色彩表示法，也就是红绿蓝三种基本颜色，类似于其它编程语言，RGB 为 16 进制数如#FFFFFF。

调色板是一个线性渐变算法，它可以平滑的将一个数映射到一个颜色。palette frac 将一个色彩乘以一个系数，系数介于 0 1 之间。palette cb 映射任意一个颜色到某种色彩。查看：set cbrange. set colorbox。

Rgb 色彩变量

plot 命令可以从文件中读取色彩信息，一个数据文件中，不同的点用不同色彩，色彩用一个整数表示。
lc rgbcolor variable 告诉 plot 命令将附加一列用于表示色彩。

```
rgb(r, g, b) = 65536 * int(r) + 256 * int(g) + int(b)
#16 进制 RRGGBB 等于 65536 * int(RR) + 256 * int(GG) + int(BB)。
splot "data" using 1:2:3:(rgb($1,$2,$3)) with points lc rgb variable
```

注意 splot 只需要三列去绘制点，这里使用了 4 列，多出的一列将用于表示色彩。

线条色彩变量

lc variable 告诉 plot 附加列用于线条色彩(lc 是 linecolor 简写)。文本色彩可以使用 tc variable 让 plot 从附加列中读取(tc 为 textcolor 简写)。

当个文件可以包括多组数据，两个空白行用于分割不同数据。pseudocolumns

```
plot ' data' using 1:2:3 with points lc variable
# Use the data set index to choose a linestyle color
plot ' data' using 1:2:(column(-2)) with lines lc variable
```

绑定

```
bind {allwindows} [<key-sequence>] ["<gnuplot commands>"]
bind <key-sequence> ""
reset bind
```

bind 命令可以定义热键，将键盘事件和脚本命令绑定到一起。在按下一个键或一系列键后执行某个脚本。
bind 命令只有当 gnuplot 编译时启用了 mouse 支持才可用。使用 show bind 命令查看当前使用的绑定。reset bind 命令恢复所有绑定到默认设置。

大部分热键只在当前获得焦点的窗口中生效。bind allwindows <key> ... (short form: bind all <key> ...)。MOUSE_KEY_WINDOW 保存了鼠标事件发生的窗口。

设置绑定：

```
bind a "replot"
bind "ctrl-a" "plot x*x"
bind "ctrl-alt-a" ' print "great"'
bind Home "set view 60,30; replot"
bind all Home ' print "This is window ",MOUSE_KEY_WINDOW'
```

显示绑定：

```
bind "ctrl-a"  
bind  
show bind
```

删除绑定：

```
bind "ctrl-alt-a" ""      #删除这个绑定。注意内建绑定无法删除。  
reset bind                #重置所有绑定到默认。  
bind!                    #同上。
```

开关绑定：

```
v=0  
bind "ctrl-r" "v=!v;if(v)set term x11 noraise; else set term x11 raise"
```

注意：修饰符名是大小写不敏感的，但是键名不是。

```
ctrl-alt-a == CtRl-alT-a  
ctrl-alt-a != ctrl-alt-A
```

不同键盘上：alt == meta

特殊键名：

```
"BackSpace", "Tab", "Linefeed", "Clear", "Return", "Pause", "Scroll_Lock",  
"Sys_Req", "Escape", "Delete", "Home", "Left", "Up", "Right", "Down",  
"PageUp", "PageDown", "End", "Begin",  
"KP_Space", "KP_Tab", "KP_Enter", "KP_F1", "KP_F2", "KP_F3", "KP_F4",  
"KP_Home", "KP_Left", "KP_Up", "KP_Right", "KP_Down", "KP_PageUp",  
"KP_PageDown", "KP_End", "KP_Begin", "KP_Insert", "KP_Delete", "KP_Equal",  
"KP_Multiply", "KP_Add", "KP_Separator", "KP_Subtract", "KP_Decimal",  
"KP_Divide",  
"KP_1" - "KP_9", "F1" - "F12"
```

下面这个是一个窗体事件名而不是一个键名。

```
"Close"
```

绑定空格

如果 gnuplot 编译时使用 `-enable-raise-console` 选项，在 plot 窗口上按下空格键可以提升

gnuplot 命令行窗口到顶端同时获得焦点。使用'gnuplot -ctrlq'命令让 ctrl-space 完成这个功能。

鼠标输入

x11, pm, windows, ggi, and wxt 终端可以接受鼠标信息。你可以将鼠标和其它脚本绑定起来。使用命令 `pause mouse` 等待鼠标事件。然后你可以使用内建变量来获取鼠标返回信息。查看：`bind ; mouse`

`variables ; set mouse`

鼠标变量

当鼠标功能开启，单击激活的 gnuplot 窗体会设置鼠标相关的用户变量。MOUSE_X

MOUSE_Y MOUSE_X2 MOUSE_Y2，如果单击鼠标时修饰键的状态保存在：

MOUSE_BUTTON MOUSE_SHIFT MOUSE_ALT MOUSE_CTRL。这些变量在绘图窗口刚刚启动时都是未定义的。只有在绘图窗口上发生单击才定义。在脚本中利用鼠标绑定和 `pause mouse` 命令结合鼠标变量可以完成一些事情如：

```
set term wxt
plot sin(x)
pause mouse
if (defined(MOUSE_BUTTON)) call ' something_else' ; \
else print "No mouse click."
```

```
set term wxt
plot sin(x)
pause mouse keypress
print "Keystroke ", MOUSE_KEY, " at ", MOUSE_X, " ", MOUSE_Y
```

鼠标事件的读取只有当鼠标事件发生在 gnuplot 生成的绘图窗口上时才能被 gnuplot 捕获。

```
pause mouse #暂停脚本执行，等待鼠标事件。
```

`pause mouse keypress` 等待键盘事件，而不是鼠标事件(鼠标按键无效果)。当键盘事件发生时同时记录下鼠标状态。MOUSE_KEY 变量包含被按下的键盘键的 ascii 值。MOUSE_CHAR 包含这个字符的字符串版本。MOUSE_X, MOUSE_Y 分别是事件发生时鼠标横纵坐标。MOUSE_X2 MOUSE_Y2 鼠标在第二坐标系的坐标。鼠标等待输入时，使用 ctrl-c 关闭了绘图窗口，MOUSE_KEY 值为-1；

GPVAL_X_MIN, GPVAL_X_MAX 绘图区域横坐标的最小值和最大值。

GPVAL_Y_MIN, GPVAL_Y_MAX 绘图区域纵坐标的最小值和最大值。

实例：

```
plot sin(x)
while(1) {
  pause mouse keypress
  print sprintf("Press %c at \n System 1 :%g,%g",MOUSE_KEY,MOUSE_X , MOUSE_Y)
  print sprintf("System 2 :%g,%g",MOUSE_X2 , MOUSE_Y2)
}
```

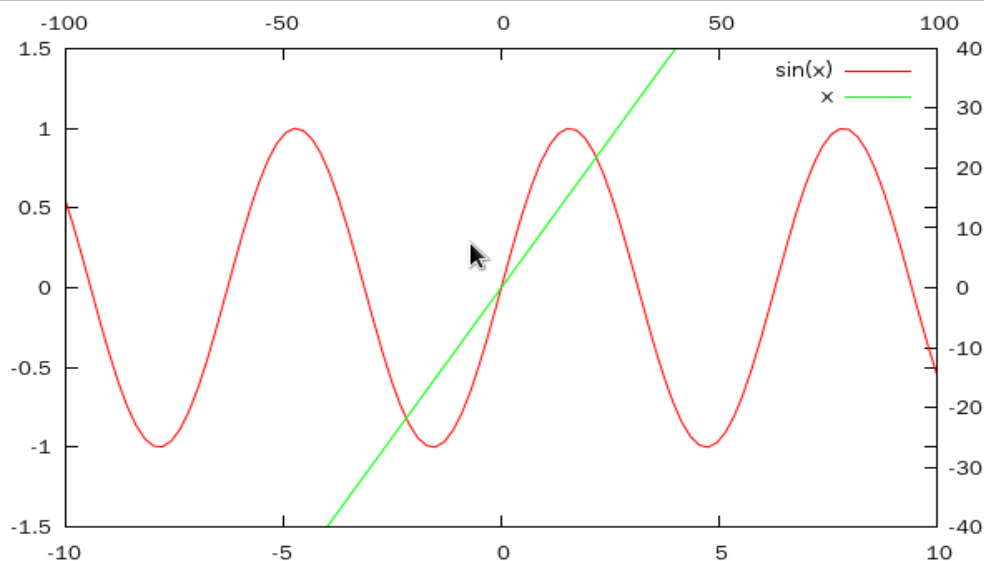
绘图命令

gnuplot 有 3 个绘图命令 plot, splot 和 replot. 。 plot 绘制二维图像，splot 绘制三维图像。replot 用于在上一个绘制命令后继续增加绘制参数。plot 可以使用 垂直坐标系和极坐标系（set polar 命令）。splot 可用于 3 维垂直坐标系利用 set mapping 命令可以使用其它坐标系（球面坐标系，柱面坐标系）。using 关键字参考前文，用于指定绘制时使用的数据。

对于一个点有四个坐标轴。x1 底横轴 y1 左纵轴 x2 顶横轴 y2 右纵轴。

plot 命令 可以在绘图时指定这个坐标轴。

```
set xrange [-10:10] #设置范围
set yrange [-1.5:1.5] #设置范围
set x2range [-100:100]
set y2range [-40:4]
set xtics;set x2tics;set ytics ;set y2tics;#显示坐标轴刻度。
plot sin(x) axis x1y1,x axis x2y2;#在一副图上绘制两个图形分别使用不同坐标系。
```



自行观察图形四周的坐标轴和代码的关系。set xlabel 可以设置坐标轴的名字。

初始化脚本

gnuplot 启动时自动搜索系统级名为 gnuplotrc 的启动脚本。这个文件的位置在编译时决定。show loadpath 命令查看当前搜索脚本的目录。随后程序搜索用户目录下名为 .gnuplotrc 的文件。在其它操作系统下配置文件名为 GNUPLLOT.INI 。

字符串常量和字符串变量

字符串常量、字符串变量、返回字符串的函数在大多数命令中都可接受。

```
four = "4"
graph4 = "Title for plot #4"
graph(n) = sprintf("Title for plot #%d",n)
plot ' data.4'      title "Title for plot #4"
plot ' data.4'      title graph4
plot ' data.4'      title "Title for plot #".four
plot ' data.4'      title graph(4)
```

他们将产生相同的标题。

数字可以自动转换成字符串所以在表达式 "four==" . 4 会生成字符串"four==4"。

主要有 3 个字符串二元运算符：".", ne eq .。见 14。

命令替换和命令行宏

命令行替换可以使用反引号`，使用反引号将扩展成命令标准输出。例如

```
str=`whoami`
```

str 内保存着系统命令 whoami 的输出。应该使用双引号括住它。否则收到提示：undefined variable: root。因为返回的字符串被作为表达式执行了。

```
set label "generated on `date +%Y-%m-%d` by `whoami`" at 1,1
```

这个命令自动设置标签为系统时间和用户名。命令替换将命令标准输出替换到原位置，就像用于输入的代码一样。

#a.txt 内容为：print "hello"

```
`cat a.txt`  
#输出 hello
```

所以使用命令替换必须小心，否则会导致意外的语法错误。

```
#a.txt 内容为 : hel"llo  
line="`cat a.txt`"
```

将导致语法错误。因为替换的结果为 `line="hel"llo"` 这是个错误的表达式。

变量替换和宏

默认情况下宏功能是关闭的，使用 `set macros` 开启它。变量替换作为宏替换就像用户键入的一样。如：

```
set macros  
style1 = "lines lt 4 lw 2"  
style2 = "points lt 3 pt 5 ps 2"  
range1 = "using 1:3"  
range2 = "using 1:5"  
plot "foo" @range1 with @style1, "bar" @range2 with @style2
```

@符号用于说明对变量进行替换作为宏。

下面这个例子让你理解宏替换和变量的区别。

```
set macros  
C = "pi"  
if (exists(C)) print C, " = ", @C  
#输出 pi=3.14159265358979
```

```
A = "c=1"  
@A  
#执行 c=1
```

但是下面这个代码不会起效果：

```
A = "c=1"; @A
```

因为他们写在同一行，宏定义同时不能扩展。如果你想执行完整的命令使用 `evaluate` 命令。如：

```
evaluate "print 'hello'"
```

字符串，宏，命令替换

字符串 宏 命令替换混合到一起有时候比较复杂。在命令替换内部可以嵌套宏。

```
set macros  
filename="file.txt"  
lines = `wc -l @filename|sed 's/ .*$/ '`
```

```
#wc -l 会在数字后输出文件名，sed 过滤掉这个文件名。  
mycomputer = " `uname -n `"
```

但是宏在双引号内就不再被解析：

```
set macros  
filename="file.txt"  
print "@filename"  
#@filename  
print "`wc -l @filename|sed 's/ .*$/`'"  
#wc: @filename: 没有那个文件或目录
```

语法

选项和选项参数由空白分割，参数值是列表或者坐标那么用逗号分割各个项目。取值范围可以使用冒号分割用方括号括住 如：[-1:1]。文本和文件名使用单引号或双引号括住。其它的一些项目使用圆括号括住。逗号可用于 set arrow、set key、set label 分割坐标，或者用于分割一个值列表（例如 fit 命令 via 关键字），离散的轮廓数据。也用 set cntrparam 命令设为循环参数，函数参数表等等。。。

圆括号用于表示一个集合。当然也可用于函数调用。

方括号用于表示取值范围，就像 plot、或 set xrange 命令中的那样，见：set xrange [-10:10]、plot [0:3][0:3] "data"。还可以用于字符串索引："string"[2:3]="tr"。

冒号用于取值范围的分割，见上文。

分号用于命令终止，类似 C 语言。

花括号用于增强文本模式，或者 if 语块，也可用于复数的表示： $\{3,2\}=3+2i$ 。

EEPIC, Imagen, Uniplex, LaTeX, and TPIC 驱动允许在单引号内使用\\表示换行，双引号内\\\\表示换行。

引号

gnuplot 使用三种引号分割文本：单引号、双引号、反引号。类似 shell，单引号中除了单引号和反斜线本身可以被反斜线转义其它反斜线都作为普通字符。双引号内会发生命令行扩展，转义序列。

```
print "first line\n second line"
```

将产生两行

```
first line  
second line
```

```
print ' first line\n second line'
```

产生：

first line\n second line

反引号用于获取系统命令的标准输出，见：25。

日期和时间

gnuplot 可以从数据文件中读取时间，实际上内部时间被储存成自 2000 年以来的秒数，如果年份在 2000 年以前，这个数值就是负数。使用 `set xdata time`, `set ydata time` 使得坐标轴上的标签为表示时间的字符串，它由当前的坐标的数值转换成时间文本。因此 **gnuplot 内部并没有时间这个数据类型，它只是一个数值**。`set timefmt` 命令用于设置时间的字符串格式，它将用于解析字符串为时间。简单来说所有输入数据必须符合这个格式。如果从文件中读取的 `x y` 字段分别表示时间和日期，那么他们也必须在一个内格式出现，例如：`set timefmt "%m/%d/%y %H:%M"`；如果你的数据中包含多种格式的时间文本，请将它们表示成字符串。这意味着如果你的数据文件中如果**有多个时间字段**，那么它们的格式应该相同。如果你的数据中包含多种时间格式，应该把它们作为字符串字段，然后使用

```
strptime ( " format" ,stringcolumn(X))
```

按照 `format` 格式解析第 `X` 字段的日期 为秒数。

gnuplot 没有提供转换时区的函数。大部分情况下你并不关心所有的数据是否处在同一时区（如果统一时区对你很重要，应该在记录时间数据时转换成 UTC 时间，而不是事后伤脑筋）。你可以手动转换时区，只需要加减时区偏移。

`show xrange` 命令会将整数数据转换成 `timefmt`。如果你改变了 `timefmt`，应该再次执行 `show`。如果没有对坐标轴激活时间格式(`set xdata`)那么会输出时间的整数格式。

`set format` 或 `set tics format` 格式设置坐标轴刻度的标签格式，它不管数据是否是时间数据。

如果 `plot` 命令的时间/日期数据从文件中读取，那么必须使用 `using` 关键字。绘图命令认为空白字符是列的分割符，但是空白可能是时间/日期数据的一部分。如果你使用 `tab` 作为列分隔符，最好进行一些测试去判断你的系统如何处理 `tab`。

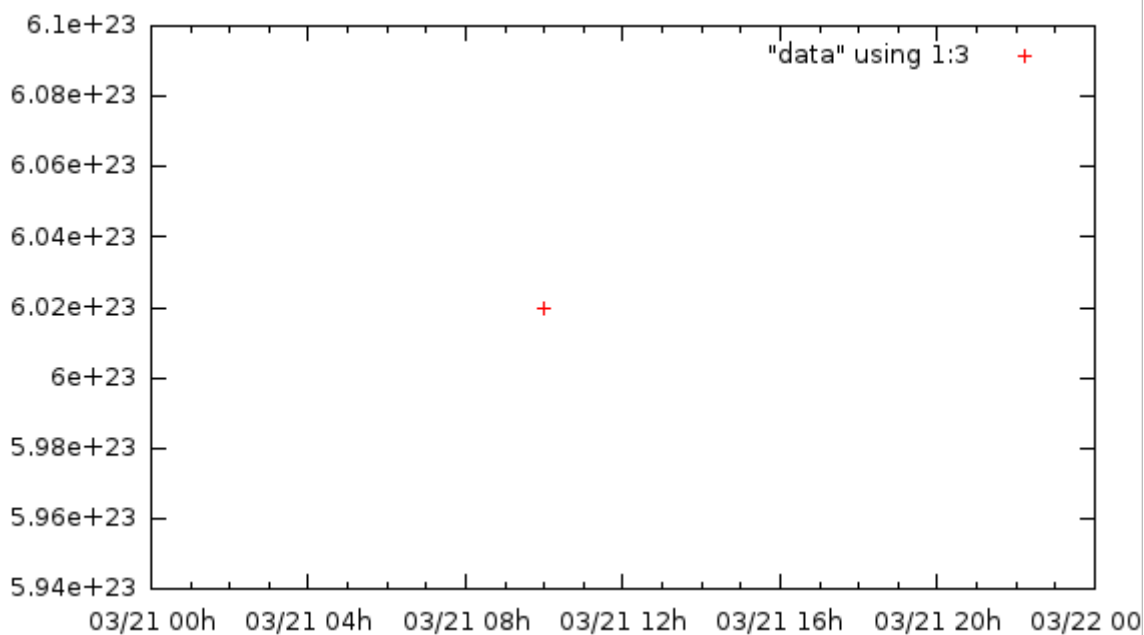
`time` 函数用于获取当前系统时间，它返回一个整数。`strftime` 函数将这个整数格式化成字符串返回。

data 文件内容：

03/21/95 10:00 6.02e23

脚本：

```
set xdata time ;set timefmt "%m/%d/%y"
set xrange ["03/21/95":"03/22/95"]
#注意，范围内使用了字符串，gnuplot 利用 timefmt 将它转换成秒数
set format x "%m/%d %Hh"
set timefmt "%m/%d/%y %H:%M" ;
plot "data" using 1:3
```



第二部分

绘画样式

gnuplot 支持很多绘画样式。set style data 和 set style function 命令修改默认的绘画样式。在 plot 或 splot 命令内使用选项指定绘画样式。如果在一条 plot 语句中绘制了多个图形，那么需要为每个图形指定绘画样式。

```
plot 'data' with boxes, sin(x) with lines
```

with 关键字后接绘画样式，boxes 是一样式，lines 是另一个样式。每个样式有自己对数据集合有不同的要求。例如 默认的 lines 样式需要一个值作为 y，使用自动生成的 x，或者使用数据库中的 x。

Boxerrorbars

它只能用于 2d 绘图。Boxerrorbars 样式是 boxes 和 yerrorbars 样式融合的产物。它需要 3, 4, 5 输入列。输入数据根据输入列¹数目不同有多种理解²：

x y ydelta

x y ydelta xdelta #set boxwidth != -2.0

- 1 输入列：原文 input columns，意为 using 关键字的参数：plot 'points.dat' using 2:3 with points。using 2:3 表示传递给绘制样式的参数。points 样式把第一个参数当作横坐标，第二个当作纵坐标。此命令含义为读取 points.dat 文件，把文件的第 2 字段用作横坐标，第 3 字段用作纵坐标，绘制点。并不是数据文件中所有列都被使用，只有被使用的列才作为输入传递给绘制样式。
- 2 根据传递给绘制样式的参数数目不同，样式对参数的用法也不同。类似函数的重载。

x y ylow yhigh #set boxwidth -2.0 , 4 个参数有两个方案, 由 boxwidth 选项决定如何选择。

x y ylow yhigh xdelta

x 横轴, y 轴值, ydelta 表示 y 正负误差范围, ylow yhigh 误差低值, 误差高值。当 boxwidth 设为-2 时使用第三个解析方案。boxwidth 设置默认矩形的宽度。

矩形的高度取决于 y 值, 线段的上下端点取决于 y 的误差值。

例子:

data:

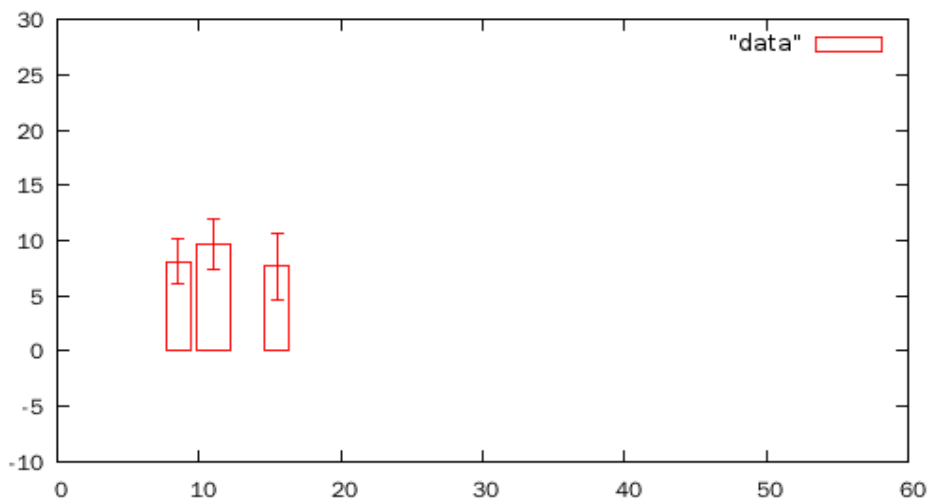
```
#x y ydelta xdelta #set boxwidth != -2.0
```

```
8.55 8.0908 2.02 1.725
```

```
11.062 9.757 2.27 2.345
```

```
15.55 7.6612 3.02 1.74
```

```
set boxwidth 2 #boxwidth 设置默认矩形的宽度。  
plot [0:60][-10:30] "data" with boxerrorbars
```



Boxes

boxes 样式只能用于 2d 绘图。它绘制直方图。可接受 2, 3 输入列。

x y

x y x_width

set boxwidth 用于设置默认 box 的宽度, box 的高度由 y 值确定。当输入数据为 3 列那么 x 宽度取第 3 列的值。如果即没有默认宽度, 也没有第 3 输入列, 每个矩形的宽度自动设置为合适宽度。

矩形内部由填充样式决定。填充样式由 set style fill 决定。plot 命令可以指定填充样式。样式取值 empty, 则不填充。

填充模式 solid 用于对矩形用当前绘制色彩实心填充。它有个参数介于 0(背景色),1(当前绘制颜色)之间。

data :

2.02 1.725

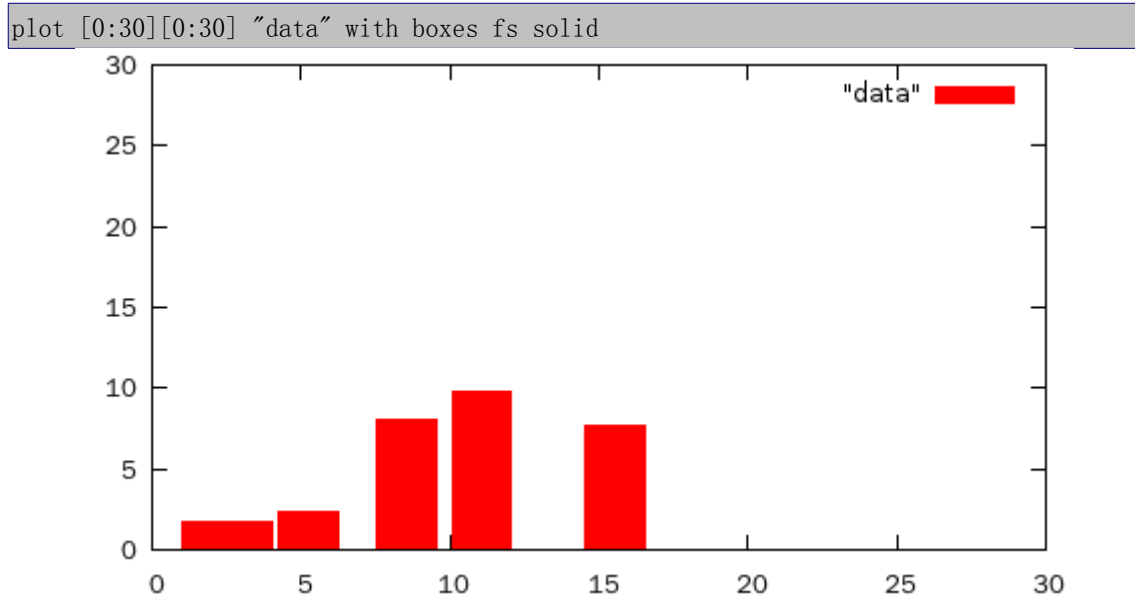
3.02 1.74

5.27 2.345

8.55 8.0908

11.062 9.757

15.55 7.6612

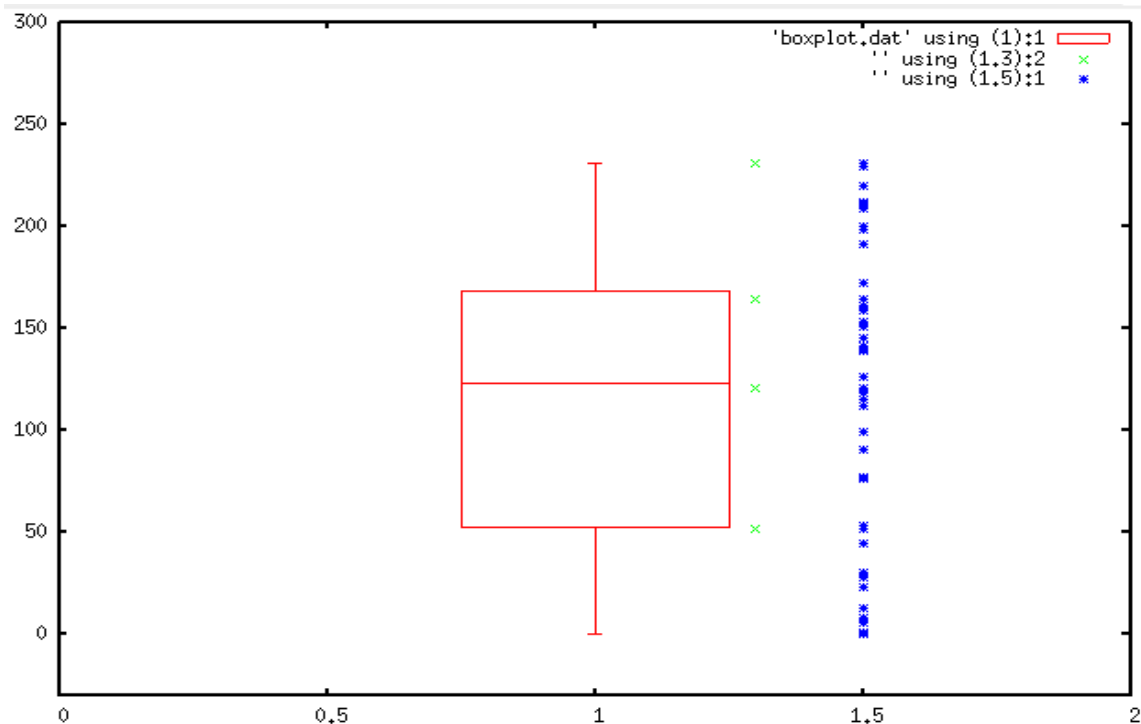


Boxplot

Boxplot 盒须图，是一种表示统计分布的通用方法。四分线由如下规则确定，第一四分位数是 $1/4$ 的点的值小于等于某个数，这个数叫做第一四分位数。 $2/4$ 的点的值小于等于某个值这个值称作第二四分位数。依次类推第三四分位数就是 $3/4$ 的值小于等于它。矩形在第一四分位和第三四分位之间绘制。并且在矩形中间绘制一条水平线。矩形上方竖直绘制一个线段到用户限制因子。在限制以外的点被独立绘制。例：

```
plot [0:2][-30:300] 'boxplot.dat' using (1):1 with boxplot
replot '' using (1.3):2, '' using (1.5):1
```

boxplot.dat 文件经过排序，数据有 40 行，第 10、20、30、40 行数据重复，他们代表第一、第二、第三、第四四分位数的值用绿色绘制。蓝色点是所有数据的图形。观察 boxplot 的横线和四分位的关系。



boxplot 可以指定矩形的宽，利用第三个参数指定。

```
plot [0:2][-30:300] 'boxplot.dat' using (1):1:(1) with boxplot
#把矩形宽设置为 1
```

当 plot 使用了三个输入列时，第一列和第三列分别作为起始横坐标和 box 宽度，最好使用常数，而不是从数据文件中读取。

```
set boxwidth <width> 也可设置矩形宽度。
```

绘制 boxplot 之间的距离默认为 1，命令 set style boxplot separation 修改它。从矩形伸出的线段尾部的横线宽度使用 set bars 命令修改。

当 using 使用了 4 个输入列那么第四输入列被当作数据标签，第一列和第三列分别作为起始横坐标和宽度处理，最好使用常数。gnuplot 会把所有拥有相同标签的数据放到一个 boxplot 中绘制，因此会产生多个 boxplot，标签会自动显示在 x 轴刻度上。boxplot 之间距离为 1，使用 separation 选项更改它。

Boxxyerrorbars

Boxxyerrorbars 只用于 2d 图像绘制。它类似于 xyerrorbars 样式，绘制一个矩形。它用 4 或 6 输入列。矩形的宽高由 x y 值以及它的误差范围所决定。

4 columns: x y xdelta ydelta

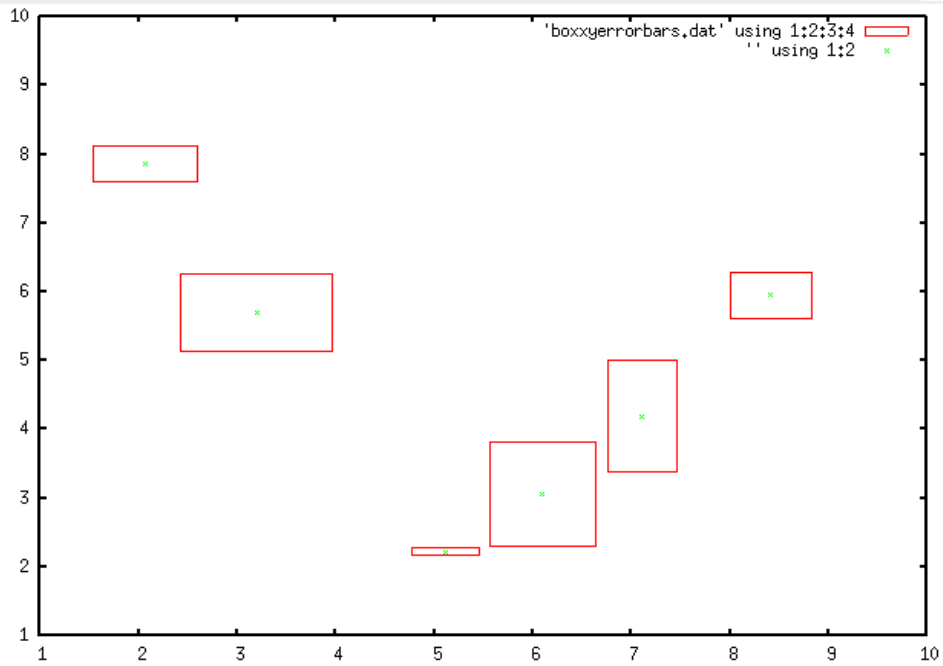
6 columns: x y xlow xhigh xlow xhigh

矩形内部填充样式由 set style fill 决定，参考 boxes 样式。

boxxyerrorbars.dat:

2.0697 7.8610 0.5245 0.2613 0.6569 0.4727
3.194 5.6995 0.765 0.570 0.126 0.8992
7.11 4.1840 0.3543 0.8109 0.1764 0.4669
8.4155 5.9487 0.416 0.332 0.1344 0.4765
5.1174 2.2232 0.3409 0.060 0.1380 0.502
6.1013 3.053 0.5349 0.7573 0.0920 0.7986

```
plot 'boxxyerrorbars.dat' using 1:2:3:4 with boxxyerrorbars  
replot '' using 1:2 #绘制 x y。观察矩形和 xy 的关系。
```



Candlesticks

蜡烛图，candlesticks 可用于经济数据（股票价格），或者为统计数据绘制统计须盒图（a box-and-whisker plot）。矩形中心横坐标为 x。矩形高度由开盘价格和收盘价格确定。高低须线的值为最高值和最低值。

financial data: date open low high close

whisker plot: x box_min whisker_min whisker_high box_high

矩形宽度由 set boxwidth 控制。为了和以前版本的 gnuplot 兼容当没有设置 boxwidth，宽度等于 set bars <width>. 的值。可以增加一个第六列，表示矩形的宽度，第六输入列格式必须和第一列(x 坐标)保持一致。

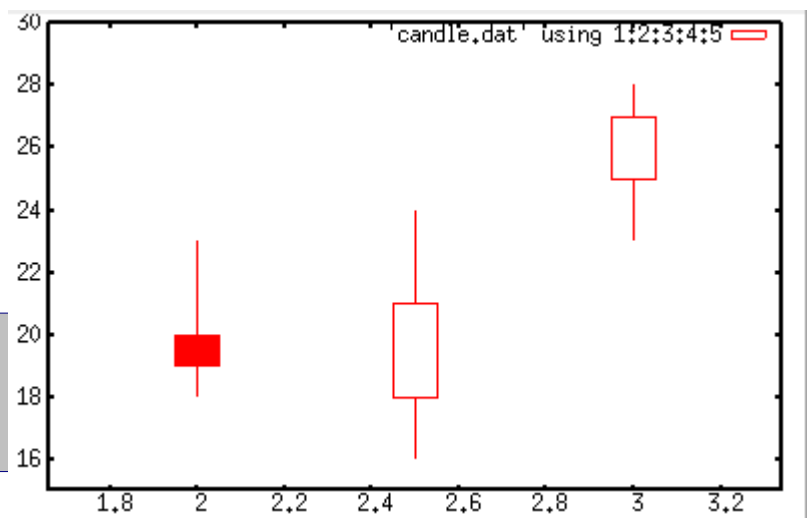
candle.dat

2 20 18 23 19

2.5 18 16 24 21

3 25 23 28 27

```
set xrange [1:10]
set yr [15:30]
plot 'candle.dat' using
1:2:3:4:5 with candle
```



candle 是 candlesticks 的简写，没有歧义就可以。也可以写成 can[dlesticks]，方括号内可以省略。

默认生成的蜡烛图须线结尾没有横线，plot 语句使用 whiskerbars 关键字将产生这个横线。当 open>close 矩形被填充。open<close 矩形空心。

Circles

Circles 样式以数据点为圆心绘制指定半径的圆。它需要 3 输入列，x y 半径。半径的单位总是和水平轴（x x2 轴）一致。默认情况下一个完整的圆被绘制出来。使用第 4 第 5 列可以指定开始角度和终止角度（角度单位是度不是弧度）。

circles.dat

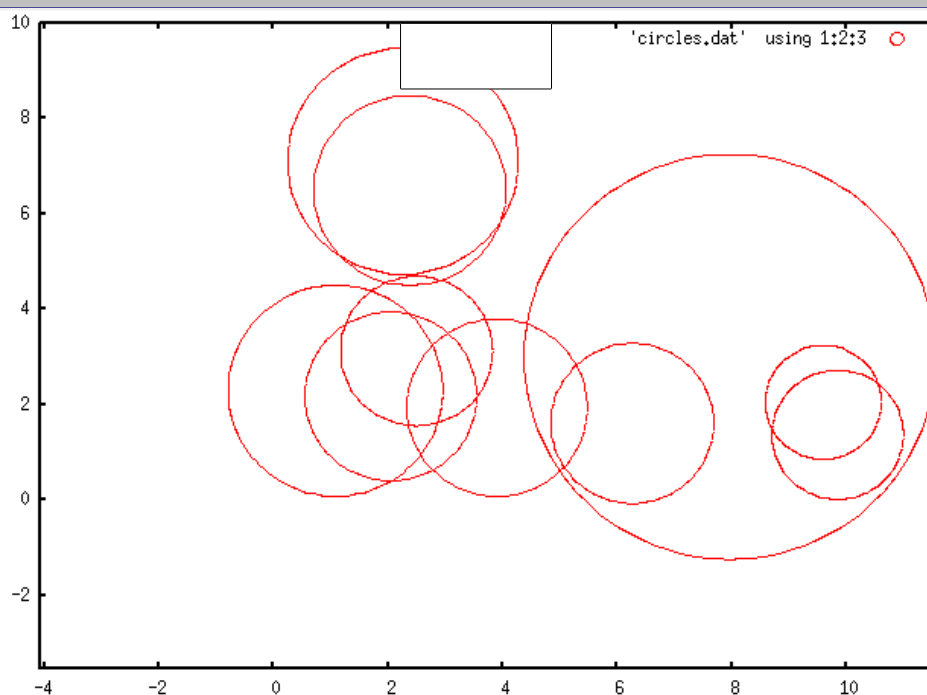
1.084 2.290 1.879 31.60 114.7

2.048 2.173 1.500 27.70 170.8

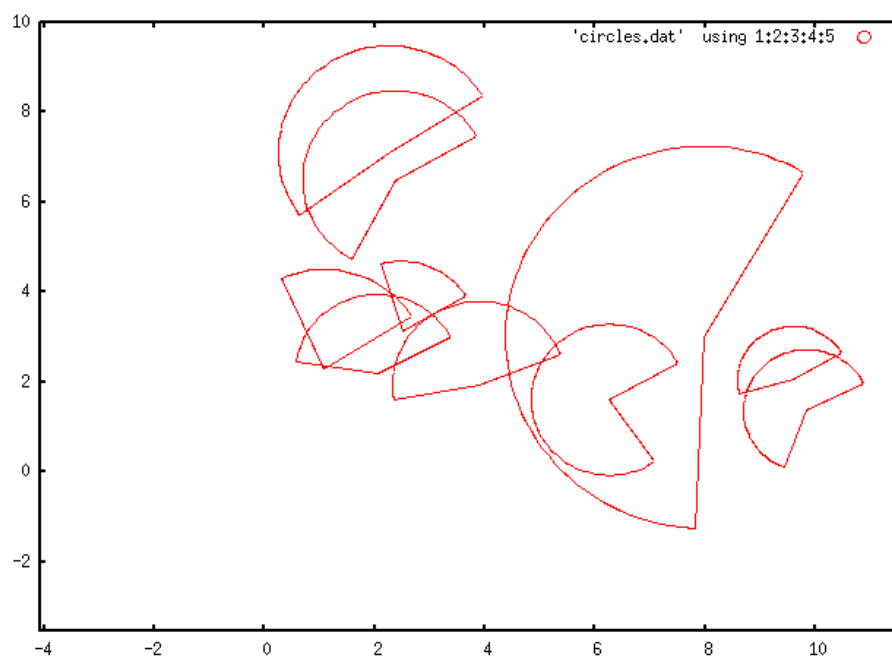
2.262 7.098 2.016 32.40 216.0

2.377 6.491 1.684 28.96 242.1
 2.499 3.119 1.330 29.66 106.5
 3.905 1.931 1.577 21.12 190.6
 6.267 1.604 1.424 28.61 305.5
 7.972 2.997 3.601 59.46 267.8
 9.601 2.041 1.012 30.08 195.3
 9.852 1.364 1.153 26.42 249.1

```
plot 'circles.dat' using 1:2:3 with circle
```



```
plot 'circles.dat' using 1:2:3:4:5 with circle
```



Ellipses

Ellipses 样式为每个点绘制椭圆。此样式只用于 2d 绘制。

接受如下输入列

2 columns: x y

3 columns: x y major_diam

4 columns: x y major_diam minor_diam

5 columns: x y major_diam minor_diam angle

当只有 x y 两输入列出现，他们表示椭圆坐标，椭圆的半径取默认值。默认值由 set style ellipse 设置。major_diam 长径 minor_diam 短径，angle 表示图形相对水平轴的偏转角度。实际上 gnuplot 把长径当作水平直径，短径当作垂直直径。水平直径单位与水平轴 (x x2) 一致，垂直直径单位与垂直轴一致。使用 3 输入列时水平径和垂直径都等于 major_diam。虽然水平径和垂直径数值相等由于椭圆两个径的单位不一样同时 xy 轴的缩放比例不同所以你看到的不一定是正圆。plot 语句使用 units 关键字指定单位。

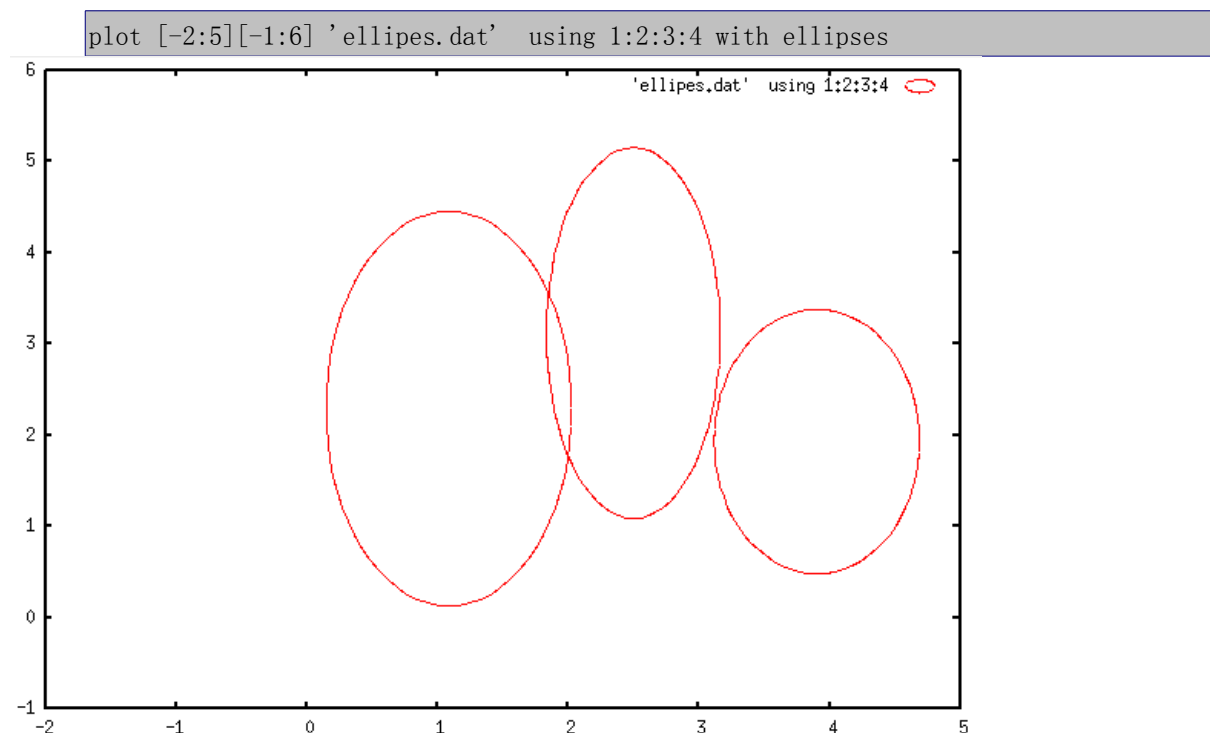
plot 语句中使用 units 关键字有 3 种格式：1. units xx 长径和短径单位与 x 轴一致。2. units yy 长径和短径单位与 y 轴一致。3. units xy 这是默认值，长轴短轴分别使用 x y 轴单位。set style ellipse. 可以控制默认值。

ellipse.dat

1.084 2.290 1.879 3.160 114.7

2.499 3.119 1.330 2.966 106.5

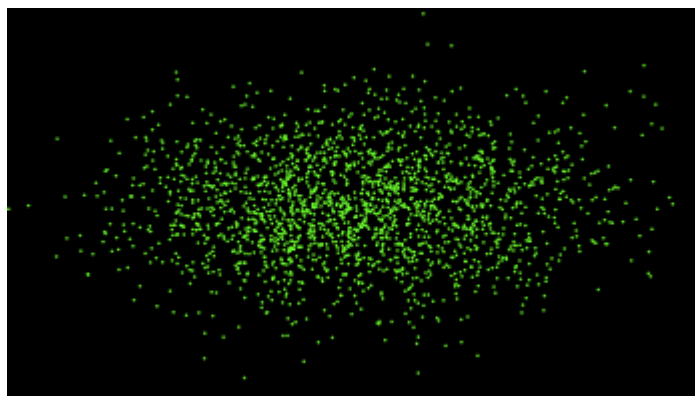
3.905 1.931 1.577 2.112 190.6



Dots

dots 样式为每个点绘制一个圆点。这对于离散绘制大量点非常有用。可以接受如下输入列：

1 column y #x 值是行号。
2 columns: x y #2D
3 columns: x y z #只用于 3D 绘图的 splot 指令



Filledcurves

Filledcurves 样式只用于 2d 绘制填充曲线。3 个变量可用。首先，前两个变量应该是一个函数或者 2 个输入列。此样式的具体行为可以通过选项更改。

语法如下：

```
plot ... with filledcurves [option]
```

选项可以是下面之一。

[closed | {above | below} {x1 | x2 | y1 | y2 | r} [= <a>] | xy=<x>,<y>]

closed 表示曲线作为封闭的多边形绘制。这是使用 2 列输入列的默认值。第二个变量表示填充区域，这个区域是曲线与指定坐标轴之间或者与某条直线之间

filledcurves closed #填充 封闭曲线

filledcurves x1 #填充与 x1 轴与曲线之间的部分

filledcurves x2 #填充与 x2 轴与曲线之间部分

filledcurves y1=0 #填充 y1=0 直线与曲线之间。

filledcurves y2=42 #填充 y2=42 直线与曲线之间。

filledcurves xy=10,20 #绘制 x1,y1 坐标系的点 10 , 20

filledcurves above r=1.5 #填充点半径 1.5 以外的部分

三个输入列： x y1 y2

填充 y1 y2 之间的区域。

fill.dat

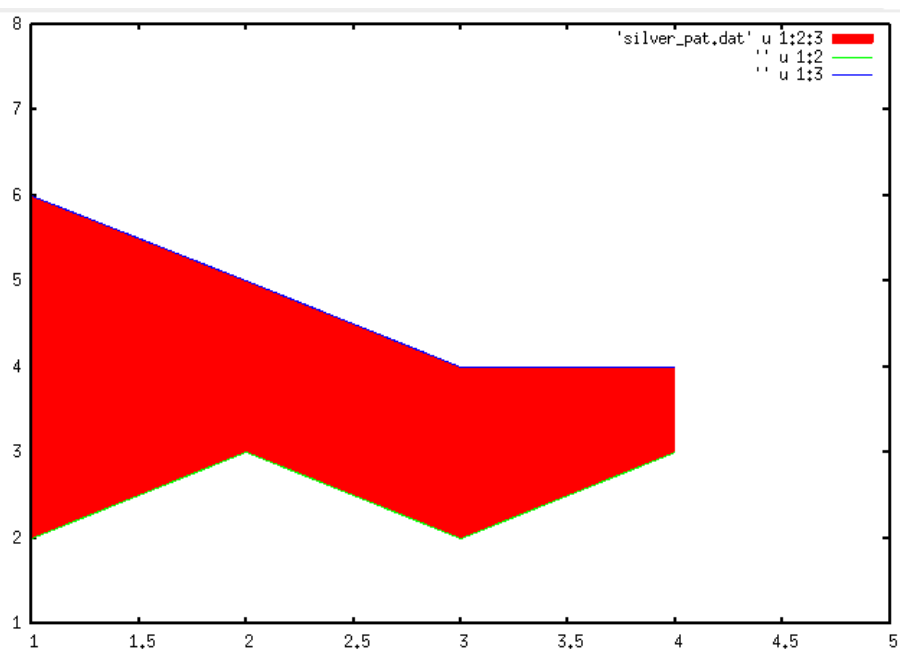
1 2 6

2 3 5

3 2 4

4 3 4

```
plot [1:5][1:8] 'fill.dat' u 1:2:3 \
w filledcu
replot '' u 1:2 w line
replot '' u 1:3 w line
```



above 和 below 语句可以使用如下格式：

... filledcurves above {x1|x2|y1|y2|r}=<val>

... using 1:2:3 with filledcurves below

这两种格式都限制了填充区域为曲线与边界线的一侧或另一侧。不是所有终端都支持这中绘制方式。

使用 xy 选项，绘制曲线起点和终点与指定点的连线与曲线之间部分。

```
plot [-3:3][-1:1] sin(x)
w filledcu xy=0.5,0.8,"
u (0.5):(0.8) w point
```

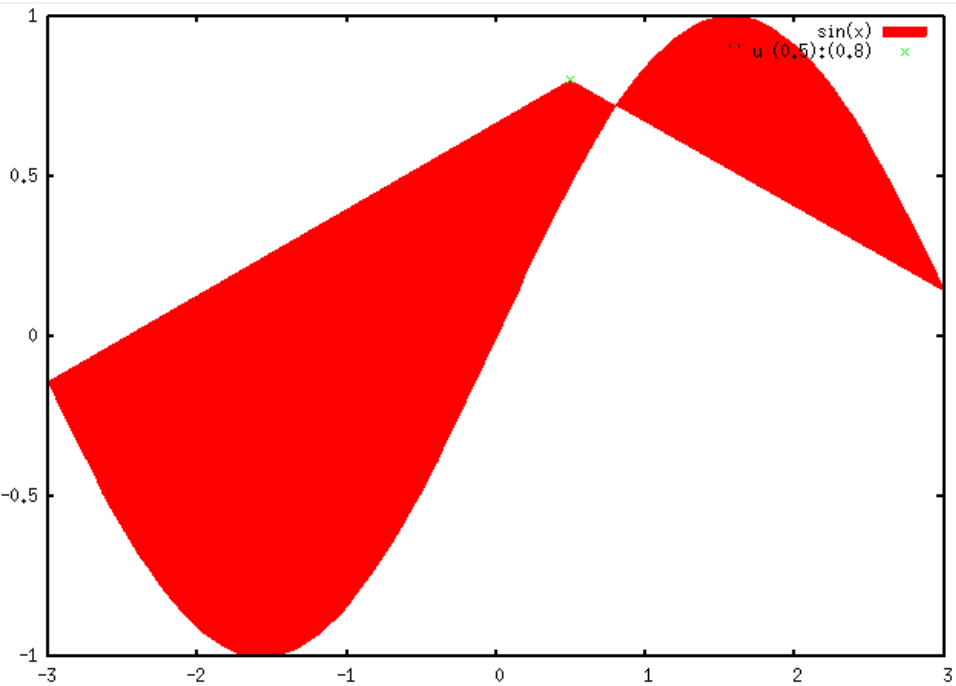
filledcurves 可读取 n 个点并且绘制曲线，然后填充曲线内部。

```
plot '-' with filledcurve
```

- 60 -80
- 55 -79
- 50 -77

可生成一个填充三角形

e



Financebars

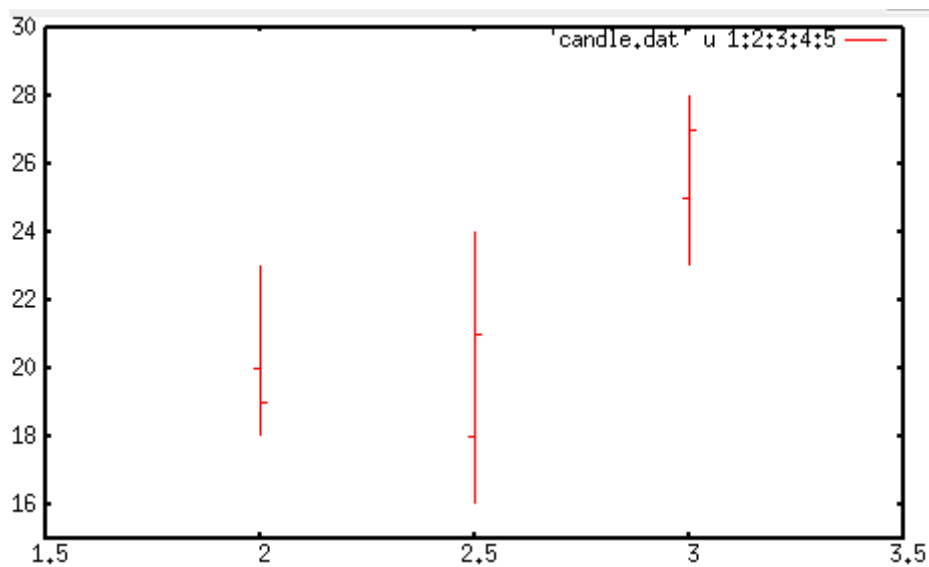
Financebars 只用于 2d 绘制，经济数据。它需要 5 个输入列：

5 columns: date open low high close

类似 candle 样式分别表示：日期 开盘价格 最低股价 最高股价 收盘价格。

符号是一个竖直线段，线段左边有个刻度表示开盘价格，右边也有个刻度表示收盘价格。见图：

```
set bars 5    #bars 宽度
plot 'candle.dat' u 1:2:3:4:5 w fin
```



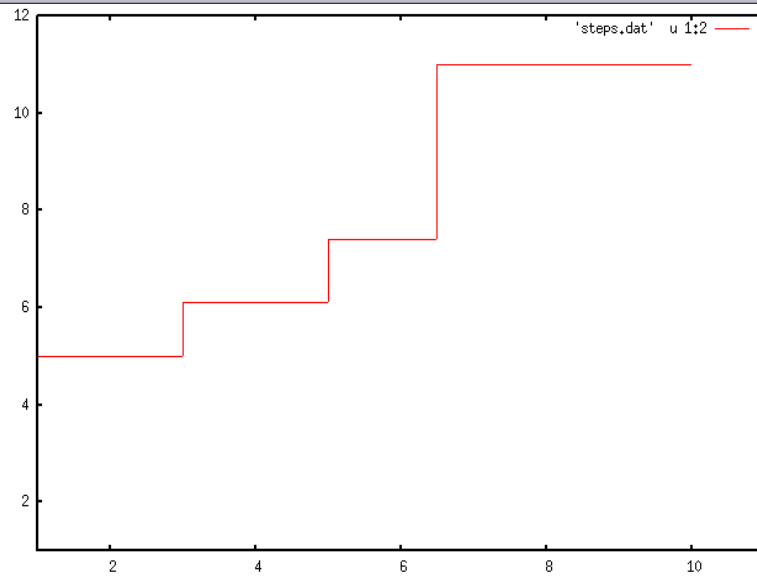
Fsteps

Fsteps 只用于 2d 绘制梯图。读取连续的 2 点，绘图分为两步：第一步 连接(x1,y1) (x1,y2)，第二步 连接(x1,y2) (x2,y2)。先绘制水平线后绘制竖直线。

steps .dat

1	2
3	5
5	6.1
6.5	7.4
10.0	11.0

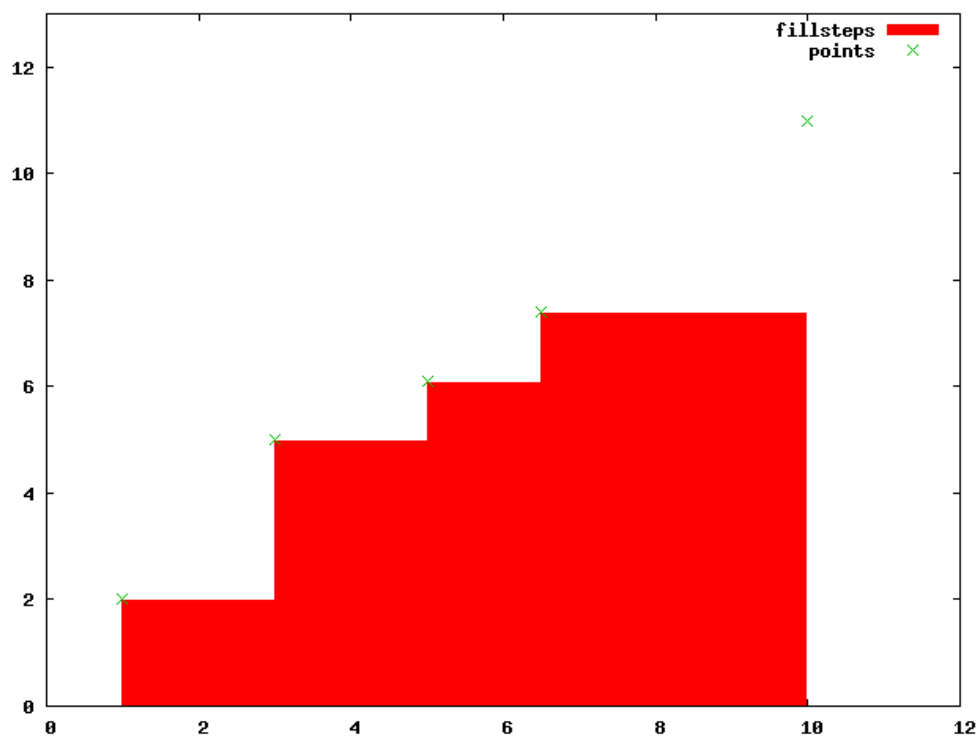

```
plot [1:11][1:12] 'steps.dat' u 1:2 w fstep
```



fillsteps

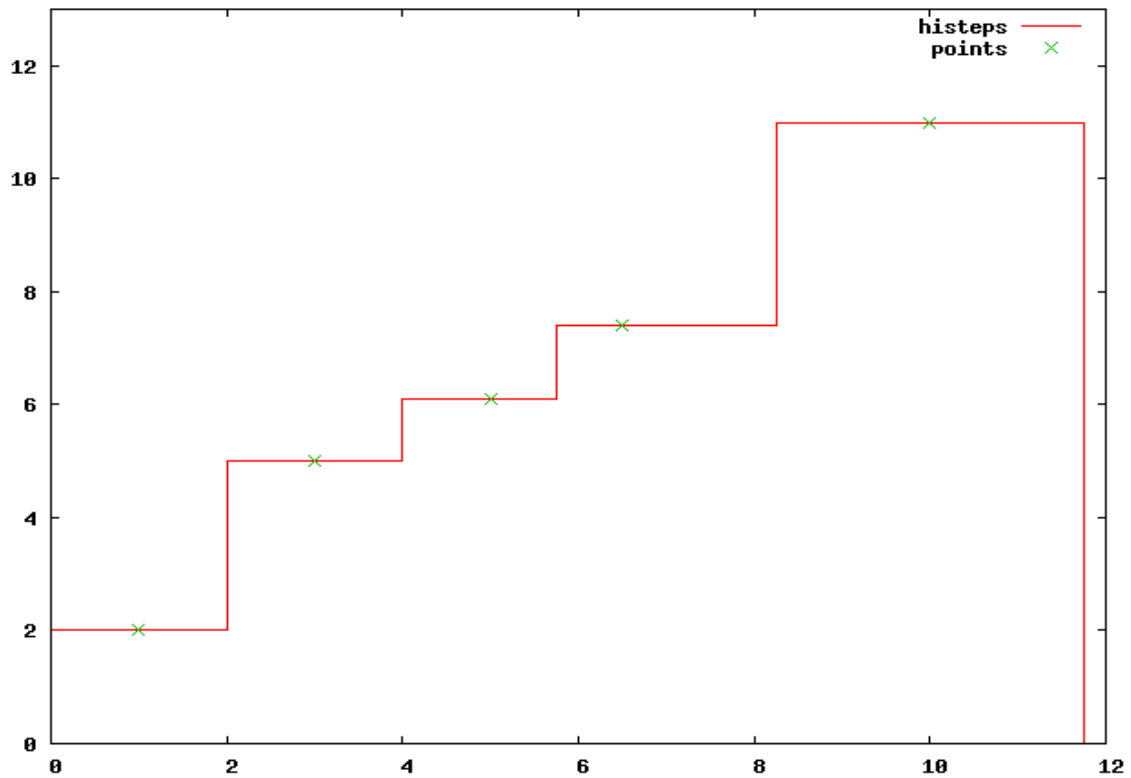
与 steps 一样，不过填充了图形与 x 轴之间部分。

```
set style fill solid 1  
plot 'steps.dat' title 'fillsteps' with fillsteps, '' title 'points' with points
```



Histeps

Histeps 只用于 2d 绘图。它用于绘制统计直方图。水平线从 $((x_0+x_1)/2, y_1)$ 到 $((x_1+x_2)/2, y_1)$ 。最后一个点的水平线段被延长，使得 x 坐标位于线段中点。竖直线段连接在 $((x_1+x_2)/2, y_1)$ 和 $((x_1+x_2)/2, y_2)$ 之间。输入列和 lines、points 样式一样。如果 autoscale 开启它根据数据文件设置 x 的范围，而不使用阶梯数目。因此最后一个点只有一半线段被显示。



Histograms

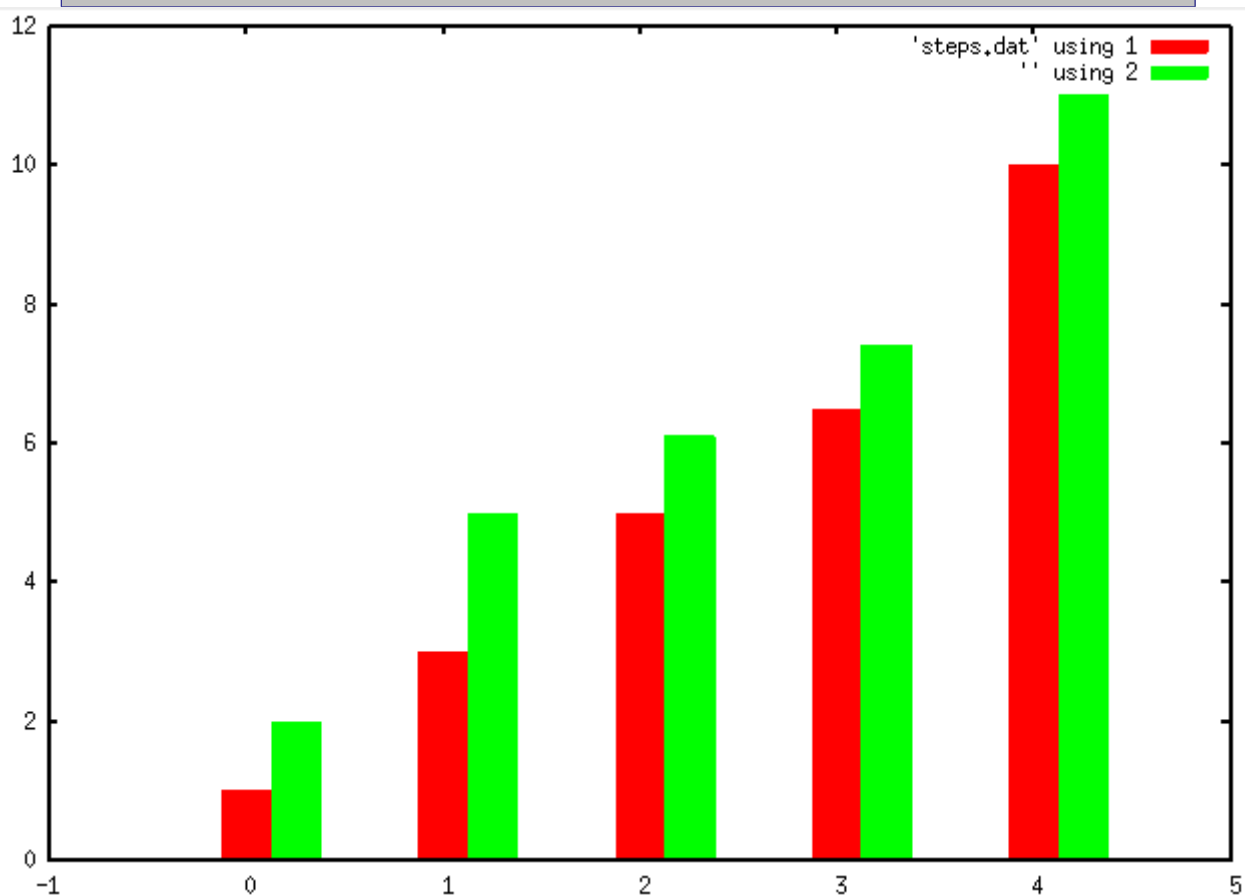
Histograms 只用于 2d 绘制。它利用多组数据生成条图。条图的各个成员必须指定成单独的输入数据（例如一个输入列），可以跟随刻度值或刻度标签。目前支持 4 中条图样式。

```
set style histogram clustered {gap <gapsize>}
set style histogram errorbars {gap <gapsize>} {linewidth}
set style histogram rowstacked
set style histogram columnstacked
```

样式默认为 `set style histogram clustered gap 2`。数据值决定图形高度，横坐标由数据所在的行序数由 gnuplot 自动排版。假设使用了 n 个输入列。第一个簇以 $x=1$ 水平居中。后接 `gap`（裂隙）然后绘制第二个簇。第二个簇与 $x=2$ 水平居中。每个簇中有 n 个矩形。`gapsize` 为 2 意味着裂隙宽等于矩形宽度 2 倍。

簇的元素为数据文件中的一行的各个字段。若需要为 x 刻度设置标签在 `using` 中使用 `xticlabels()`。

```
set style histogram clustered gap 2
set style fill solid 1
plot 'step.dat' using 1 with histogram, '' using 2 with histogram
## 注意它只能使用一个输入列。
```

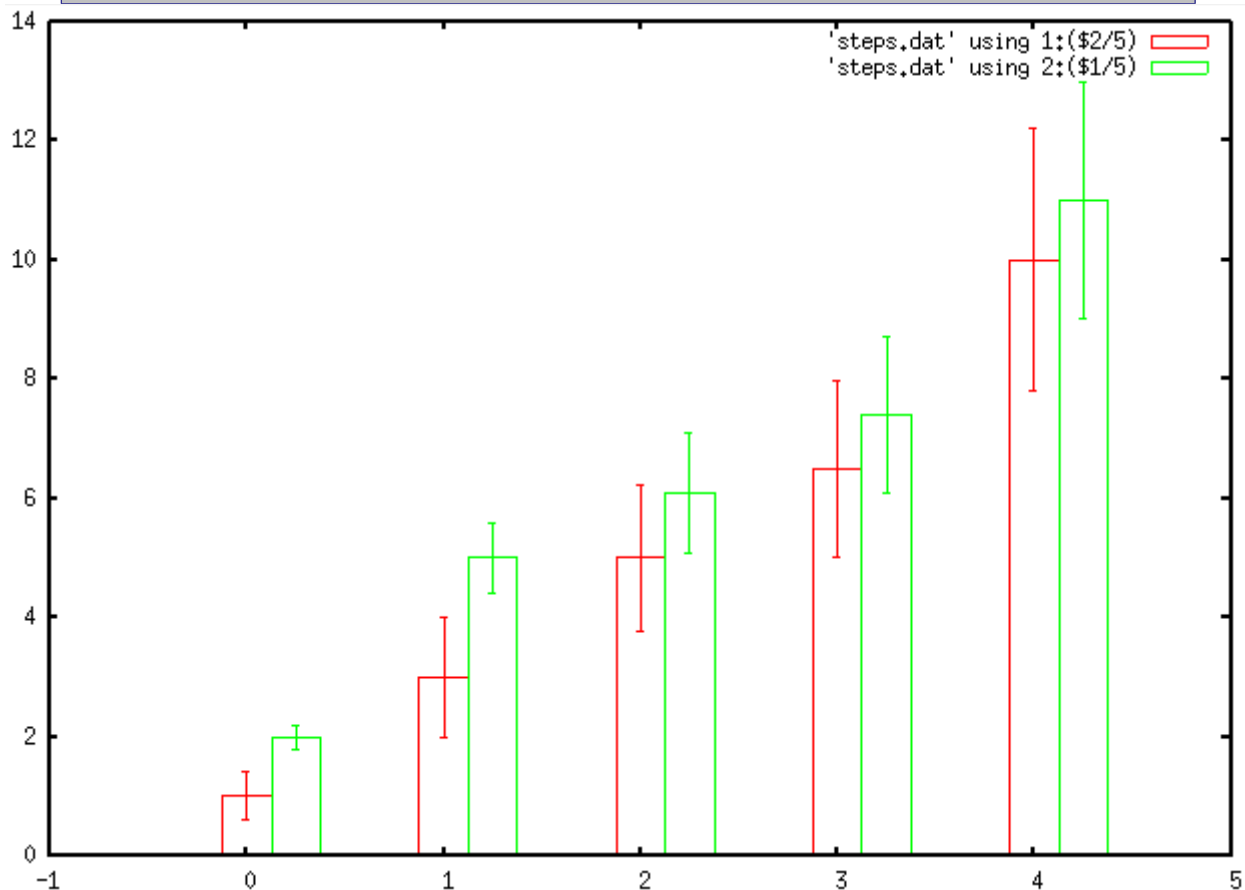


errorbars 样式与 clustered 样式相似。它需要更多的输入列。

2 columns: y yerr 须线 y-yerr 到 y+yerr

3 columns: y ymin yman 须线 ymin 到 ymax

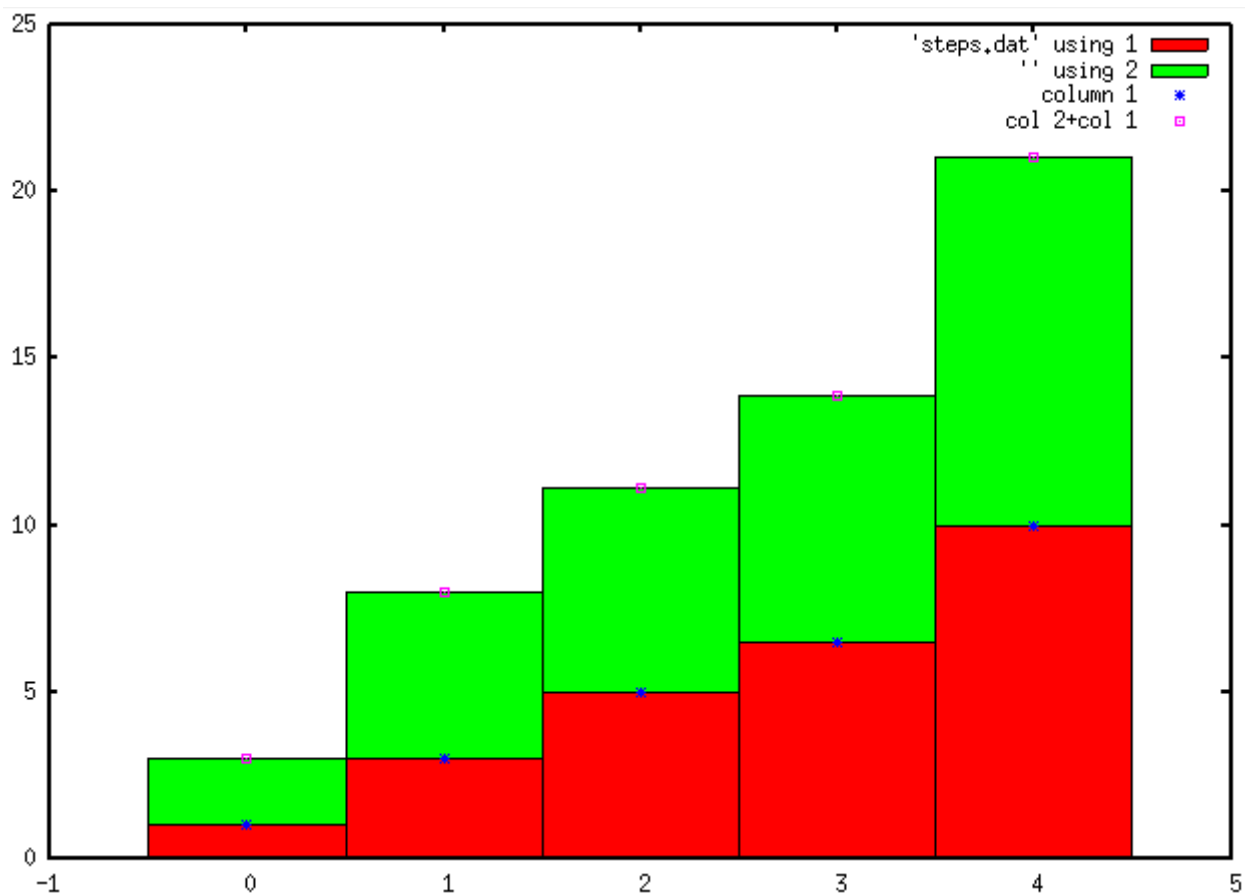
```
set style histogram errorbars gap 2
plot 'steps.dat' using 1:($2/5) with histogram
replot '' using 2:($1/5) with histogram
```



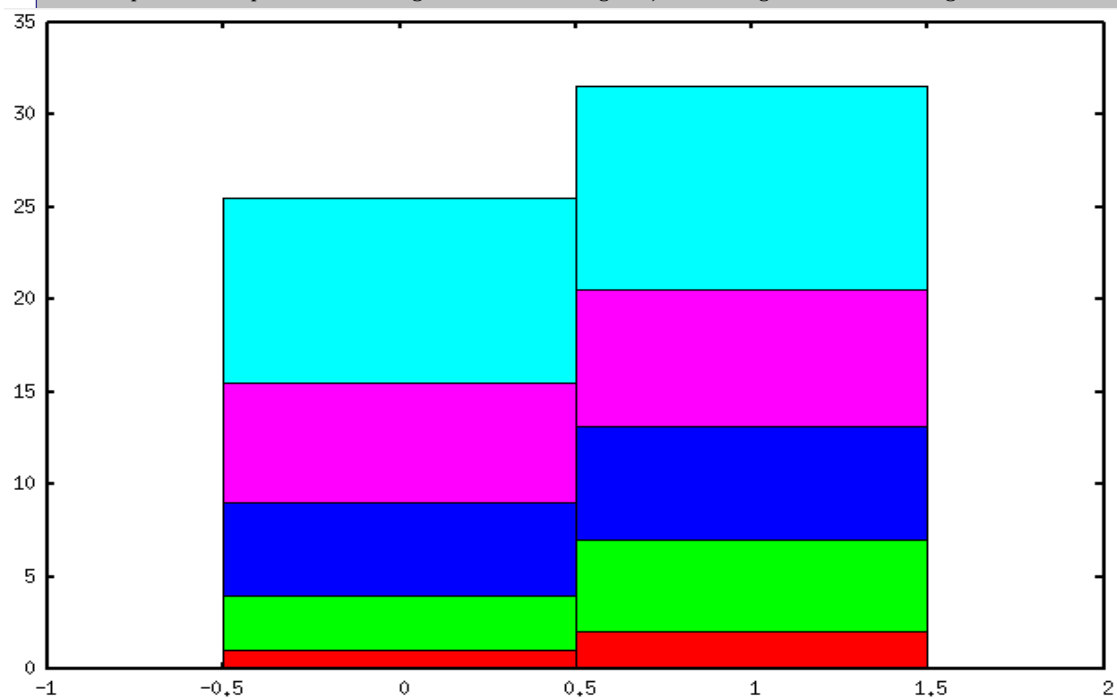
须线末端横线宽度由 set bar 命令控制。

histogram 样式支持两种堆积样式。由命令 set style histogram {rowstacked|columnstacked} 设置。使用 rowstacked，输入列的值绘制的矩形被叠加在一起。

```
set style fill solid 1 border lt -1
set style histogram rowstacked #按行堆叠
plot 'steps.dat' using 1 with histogram, '' using 2 with histogram, \
'' using :($1) title 'column 1' with points, \
'' using :($1+$2) title 'col 2+col 1' with points
```



```
set style fill solid 1 border lt -1
set style histogram columnstack #按列堆叠
plot 'steps.dat' using 1 with histogram, '' using 2 with histogram
```



Newhistogram

语法：

```
newhistogram {"<title>"} {lt <linetype>} {fs <fillstyle>} {at <x-coord>}
```

单个 plot 语句中出现多个数据集。这种情况下可以手动指定簇之间的间隙。可以为每个数据集指定单独的标签。

```
set style histogram cluster
plot newhistogram "Set A", ' a' using 1, ' ' using 2, ' ' using 3, \
newhistogram "Set B", ' b' using 1, ' ' using 2, ' ' using 3
```

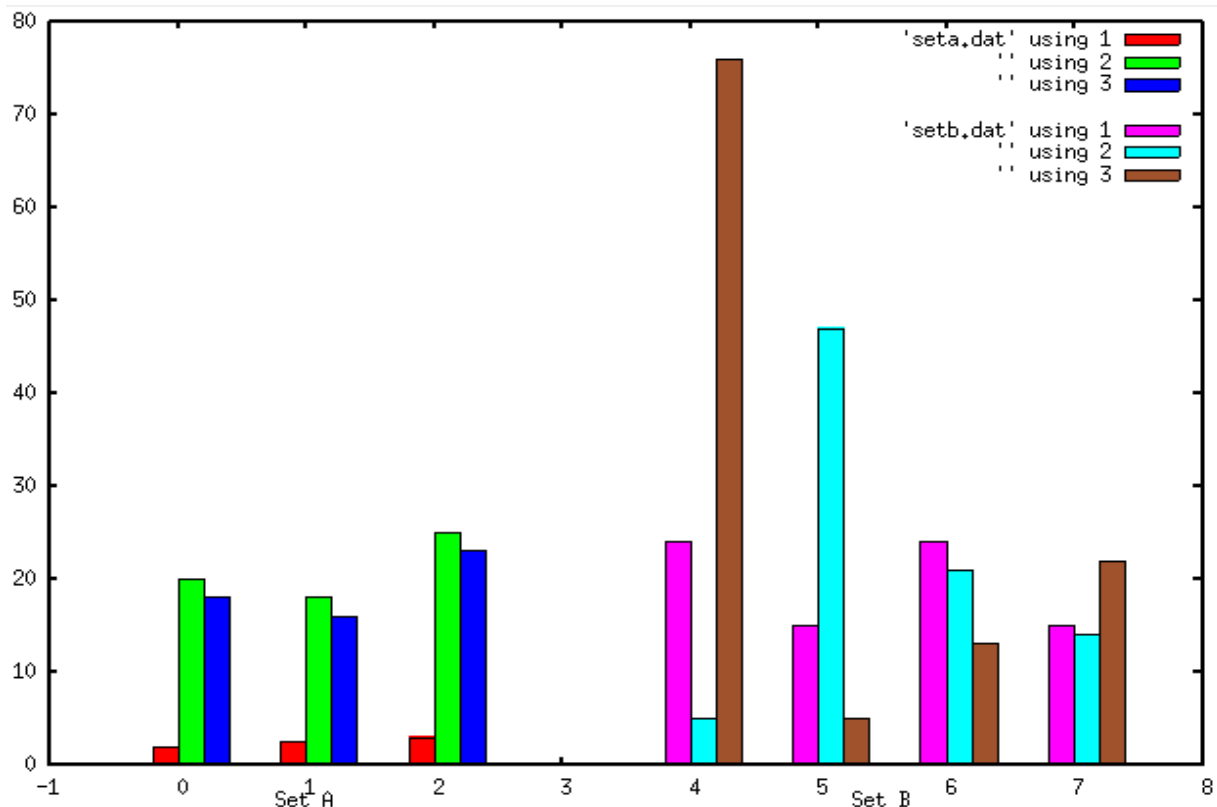
"Set A" "Set B" 标签出现在簇下方。newhistogram 命令可以指定 linetype。例如。

```
plot newhistogram "Set A" lt 4, ' seta.dat' using 1, ' ' using 2, ' ' using 3, \
newhistogram "Set B" lt 4, ' setb.dat' using 1, ' ' using 2, ' ' using 3
```

at <x-coord> 子命令用于设置起始横坐标。

```
set style histogram cluster
set style data histogram
plot newhistogram 'Set A' lt 1, 'seta.dat' using 1 t 1, ' ' using 2 t 2, ' ' using 3 t 3
replot newhistogram 'Set B' lt 4 at 4, 'setb.dat' using 1 t 1, \
' ' using 2 t 2, ' ' using 3 t 3
```

#注意 lt 选项 如果你希望两个数据集颜色一致那么设置 lt 选项为相同值。



在多列中自动递增

```
set style histogram columnstacked
plot for [i=3:8] "datafile" using i title columnhead
```

循环，i 从 3 到 8 plot 为每个 i 绘制一个图形。

Image

image、rgbimage、rgbalpha 绘制样式，都为数据产生均匀填充的方格图，用于 2d 或 3d 绘制。输入数据可以是真实的位图，也可以是能转换成为位图的其他格式如 png，或者是一个数列。

plot '-' matrix with
image

5 4 3 1 0

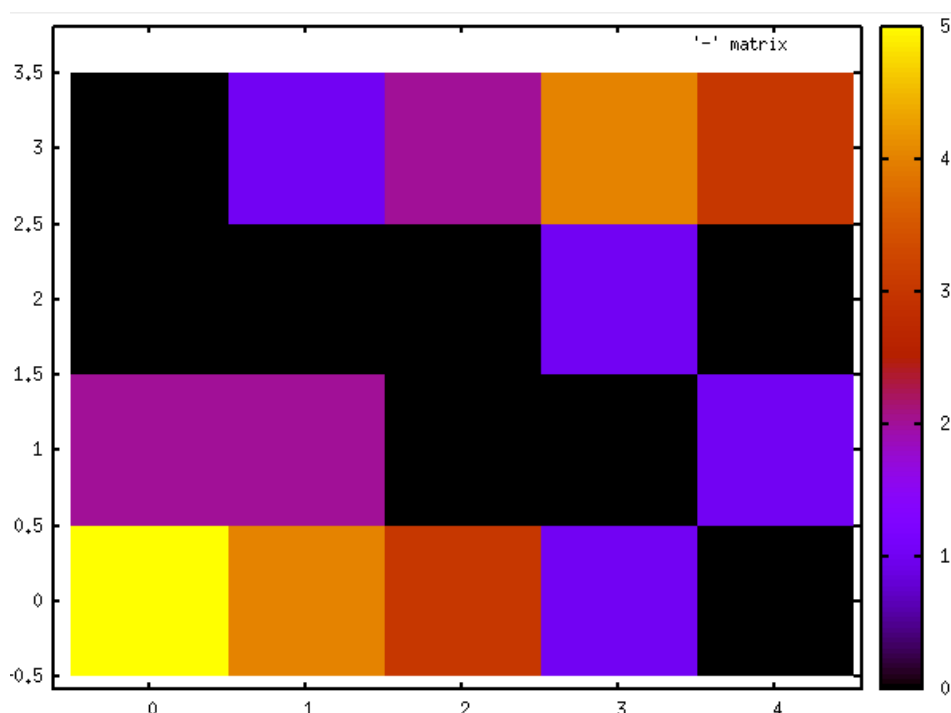
2 2 0 0 1

0 0 0 1 0

0 1 2 4 3

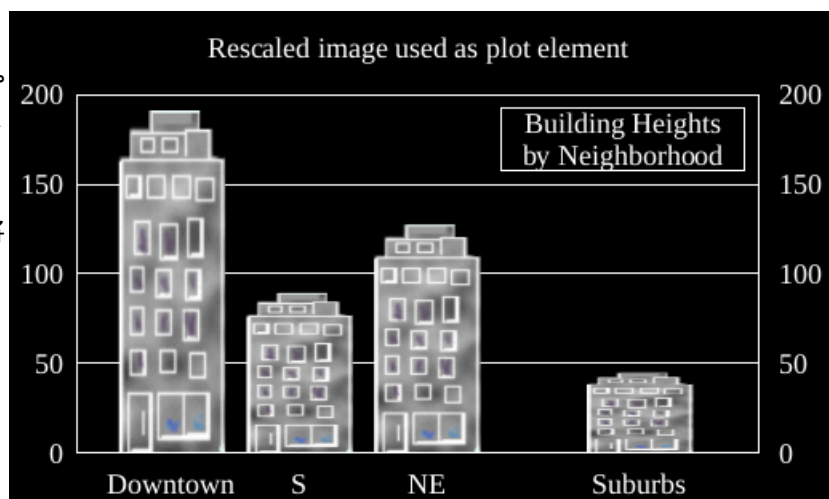
e

e



输入图形数据中的每个像素在绘制中自动转换成矩形或者平行六面体。坐标由平行六面体中心决定。 $M \times N$

矩阵将绘制 $M \times N$ 像素。在 pm3d 绘制样式中 $M \times N$ 矩阵将产生 $M-1 \times N-1$ 个像素。二进制图像的扫描方向可以被附加关键字控制：binary keywords flipx、keywords center、



keywords rotate.

图像数据可被缩放用于填充一个矩形。在 2d 坐标系中指定每个像素的 x y 扩展，参考：binary keyword dx 和 dy。原始图像 50x128 像素。较高大的建筑由 dx=0.5 dy=1.5 生成，矮小的由 dx=0.5 dy=0.35 产生。image 处理数据可接受灰阶和调色板。plot 语句使用三个输入列分别表示 x,y,value。splot 语句使用 4 个输入列分别表示 x y z value

rgbimage 将数据文件解读成由 3 rgb 色彩描述的数据。plot 使用 5 个输入列 x y rr gg bb。splot 使用 6 个输入列 x y z rr gg bb，rgb 数据值介于 0-255。

rgbalpha 处理数据包括了 alpha 通道。plot 使用 6 个输入列 x y rr gg bb alpha。splot 使用 7 个输入列 x y z rr gg bb alpha。alpha 介于 0-255。

透明度

rgbalpha 样式的 alpha 值为 0 时表示完全透明，255 表示完全不透明。当 alpha 介于 0 255 之间表示部分透明，只有少数终端类型能够正确处理 alpha 通道。其它终端类型会把它当作 0 或者 255 之一。

image 样式容错

某些终端类型提供了渲染优化。但某些情况下他们会引起问题使用 failsafe 关键字关闭渲染优化。

```
plot 'data' with image failsafe
```

Impulses

从纵坐标 0 开始绘制竖直线段。

1 column: y

2 column: x y # 线段从 [x,0] 到 [x,y] (2D)

3 columns: x y z # [x,y,0] 到 [x,y,z]

Labels

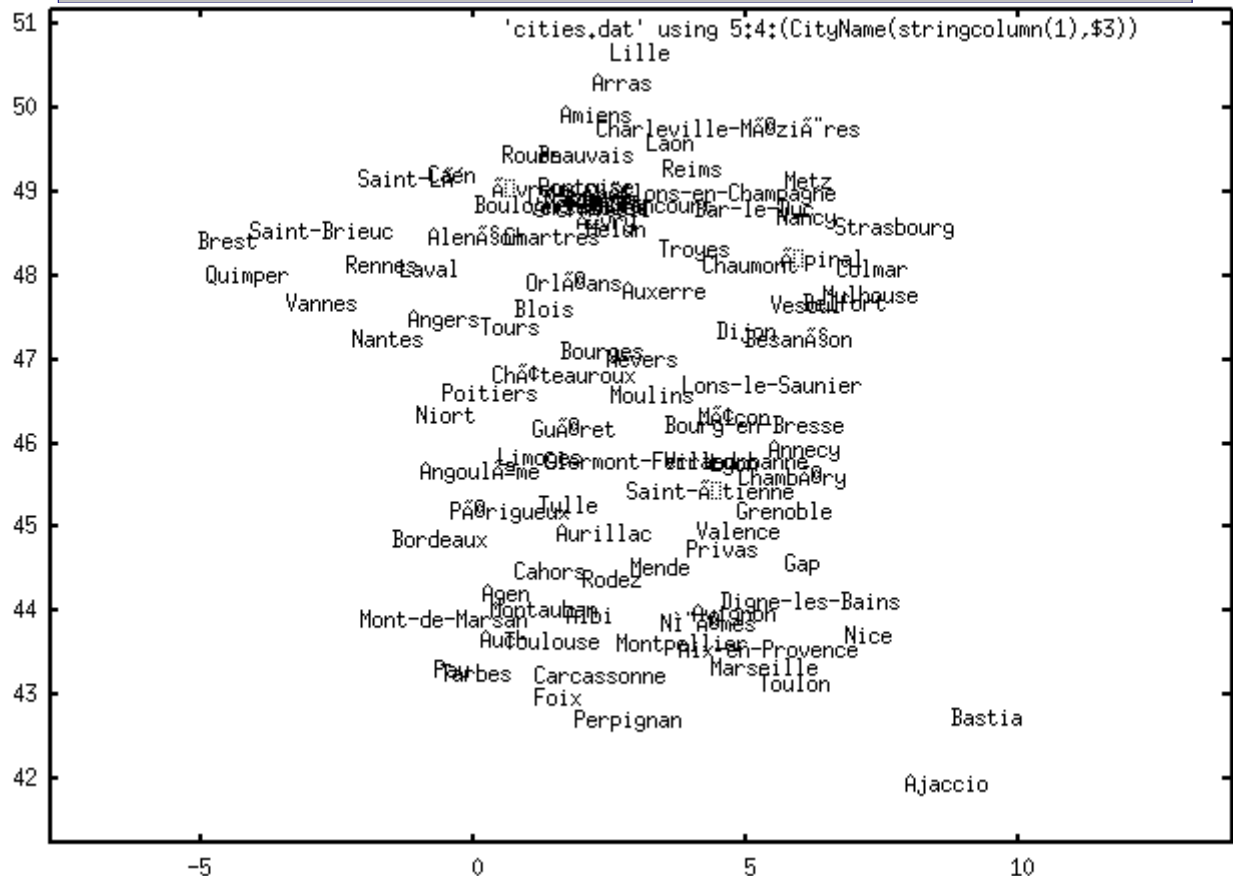
在指定坐标绘制一个指定的标签。使用 3 或者 4 个输入列。使用 rgbcolor variable 关键字从文件中读取色彩。字体色彩 旋转角度 和其它属性，可在 plot 语句中利用关键字指定。

3 columns: x y string # 2D version

4 columns: x y z string # 3D version

```
CityName(String,Size) = sprintf("{/=%d %s}", Size, String)
```

```
set termoption enhanced
plot 'cities.dat' using 5:4:(CityName(stringcolumn(1),$3)) with labels
```



Lines

依次连接数据点。

2D

1 column: y

#默认横坐标为数据所在行行号

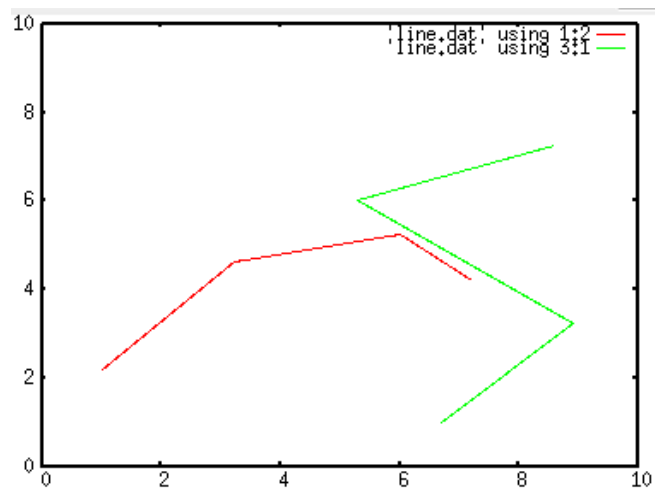
2 columns: x y

3D

1 column: z

#默认 x 为数据所在行行号, y 为索引号。

3 columns: x y z



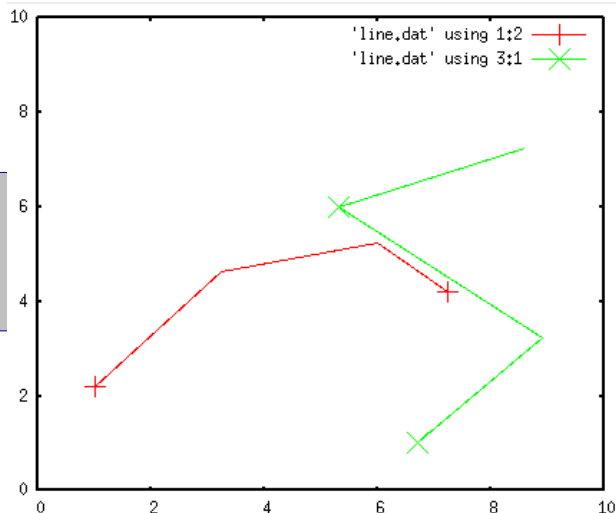
```
plot [0:10][0:10] 'line.dat' using 1:2 with lines
replot 'line.dat' using 3:1 with lines
```

Linespoints

是 lines 和 points 融合的产物。同 lines 一样依次连接各个点，然后为每个点绘制图形。set pointsize 命令改变点的半径。2d 绘制接受 1 或者 2 个输入列，3d 绘制接受 1 或者 3 个输入列。

pointinterval 属性（简写为 pi）用于控制是否每个点都被绘制。例如 'with lp pi 3' 每 3 个点绘制图形（被绘制的点的序号为 $1+n*3$ ，第 1 第 4 第 7 ...）。

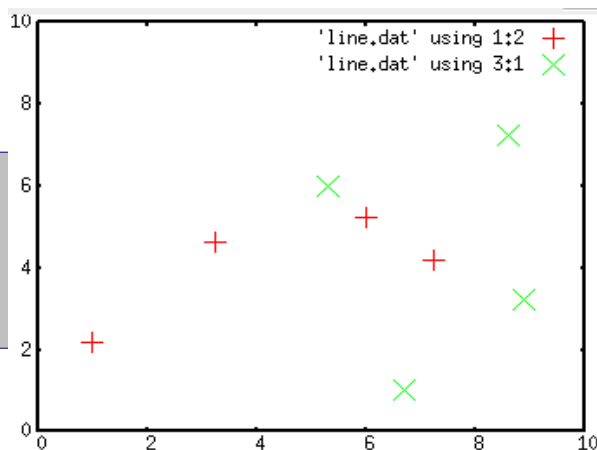
```
set pointsize 5
plot [0:10][0:10] 'line.dat' using 1:2 with
linesp pi 3
replot 'line.dat' using 3:1 with linesp pi 2
```



Points

points 样式为每个数据点绘制一个符号。set pointsize 用于修改点的大小。2d 绘制中接受 1 或者 2 输入列。3d 绘制中接受 1 或者 3 输入列。

```
set pointsize 5
plot [0:10][0:10] 'line.dat' using 1:2 \
with point
replot 'line.dat' using 3:1 with point \
point
```

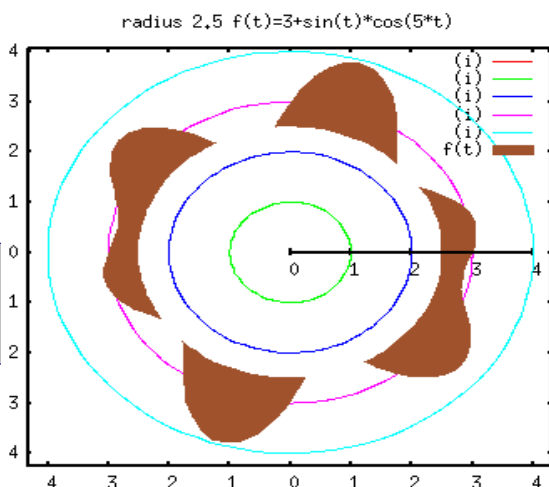


Polar

polar 不是真的绘画样式。不过它影响绘画样式。set polar 命令，使得 gnuplot 将坐标理解为极坐标系。

即<角度,半径>。大多数 2d 绘图样式可以在极坐标系下工作。左边的图形显示了 lines filledcurves 在极坐标系下的例子。

```
f(t)=3+sin(t)*cos(5*t)
plot for [i=0:4] (i) with lines,f(t) \
with filledcurve above r=2.5
```



steps

只用于 2d 绘图。它用水平线段和竖直线段连接相邻点。线段为 $(x1,y1)-(x2,y1)$ 和 $(x2,y1)-(x2,y2)$ 。接受的输入列和 `lines points` 一致。 `fsteps` `steps` 之间的不同点在于 `fsteps` 首先绘制竖直线，`steps` 首先绘制水平线。填充曲线于 x 轴相交的部分，使用 `fillsteps`。

vectors

`vectors` 样式 2d 向量绘制一个箭头，从 (x,y) 指向 $(x+xdelta,y+ydelta)$ 。3d 向量样式类似，需要 6 个输入列。箭头指向向量终点。

4 columns: x y xdelta ydelta

6 columns: x y z xdelta ydelta zdelta

一个附加列可用于色彩指定 (`linecolor`、`rgbcolor variable`) 。

`splot` 命令中向量只在 `set mapping cartesian` 命令开启后有效。

在 `with vectors` 后可跟随一些关键字用于设置箭头的样式，或者为每个向量从文件中读取自己的样式。注意：如果你使用 '`arrowstyle variable`' 它将在每个向量被绘制时单独应用属性。你不能在其它地方使用这个关键字，也不能与其它向量样式修饰关键字混合使用。

```
plot ... with vectors filled heads
plot ... with vectors arrowstyle 3
plot ... using 1:2:3:4:5 with vectors arrowstyle variable
```

查阅：`arrowstyle`。

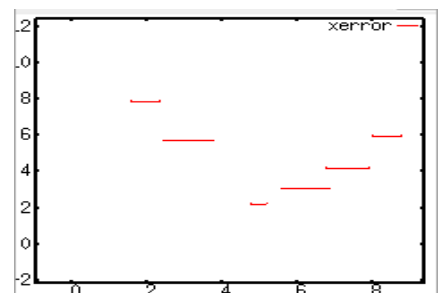
例子：

```
plot 'dat' using 1:2:3:4 with vectors head filled lt 2
splot 'file.dat' using 1:2:3:(1):(1):(1) with vectors filled head lw 2
```

`set clip one` `set clip tow` 影响向量绘制。

xerrorbars

`xerrorbars` 样式值用于 2d 绘制。`xerrorbars` 与 `points` 相似，它绘制一个水平误差条。误差条从 $(xlow,y)$ to $(xhigh,y)$ 或者 $(x-xdelta,y)$ to $(x+xdelta,y)$ 。误差条两端显示着短线(bar)，使用 `set bar` 控制端点的样式。



3 columns: x y xdelta
 4 columns: x y xlow xhigh

一个附加列可用于设置项目颜色。

```
plot 'boxxyerrorbars.dat' using 1:2:($1-$3):($1+$4) \
with xerror
```

xyerrorbars

xyerrorbars 只用于 2d 绘制。与 xerrorbars 类似，多了竖直方向误差。线段由(xlow,y) to (xhigh,y),(x,ylow) to (x,yhigh)。

4 columns: x y xdelta ydelta
 6 columns: x y xlow xhigh ylow yhigh

如果数据以混合格式给出，在 plot 中使用 using 关键字很有用例如：数据格式为 x, y, xdelta, ylow, yhigh。可以用如下命令：

```
plot 'data' using 1:2:($1-$3):($1+$3):4:5 with \
xyerrorbars
```

一个附加列可用于提供色彩。yerrorbars

yerrorbars (或者 errorbars)样式只用于 2d 绘制。与 xerrorbars 类似，yerrorbars 绘制一个竖直误差条。线段从 (x,y-ydelta) to (x,y+ydelta) 或者从 (x,ylow) to (x,yhigh)。set bar 命令用于更改端点样式。

附属列可用于提供色彩信息。

xerrorlines

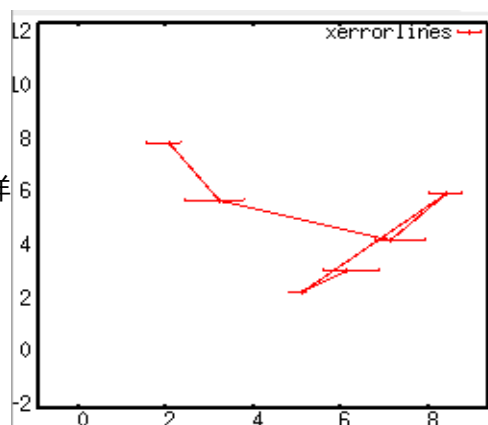
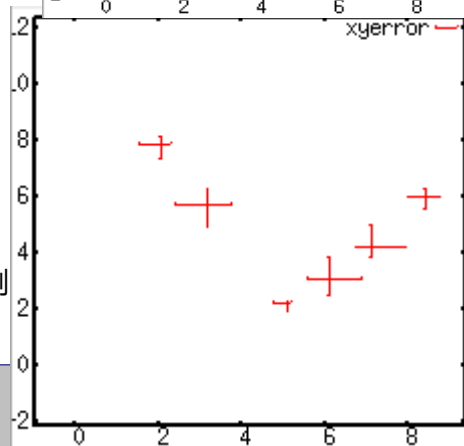
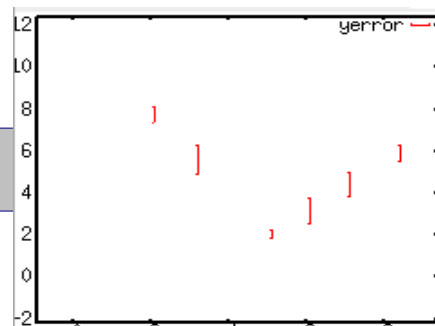
xerrorline 只用于 2d 绘制。它是 xerrorbars 样式和 lines 样式结合的产物。

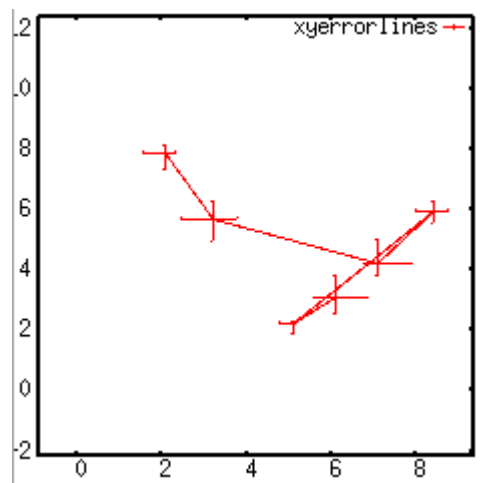
4 columns: x y xdelta ydelta
 6 columns: x y xlow ylow yhigh

xyerrorlines

xyerrorline 只用于 2d 绘制。它是 xyerrorbars 和 lines 样式混合产物。

4 columns: x y xdelta ydelta
 6 columns: x y xlow xhigh ylow yhigh





xyerrorlines

它是 yerrorbars 和 lines 结合产物。

3 columns: x y xdelta

4 columns: x y ylow yhigh

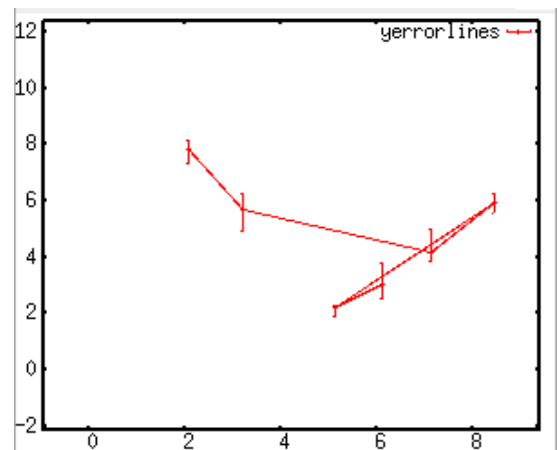
3D 面绘制

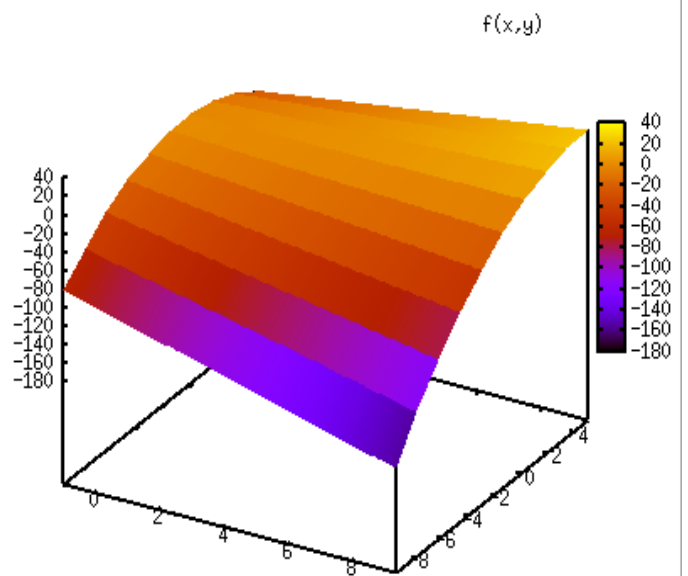
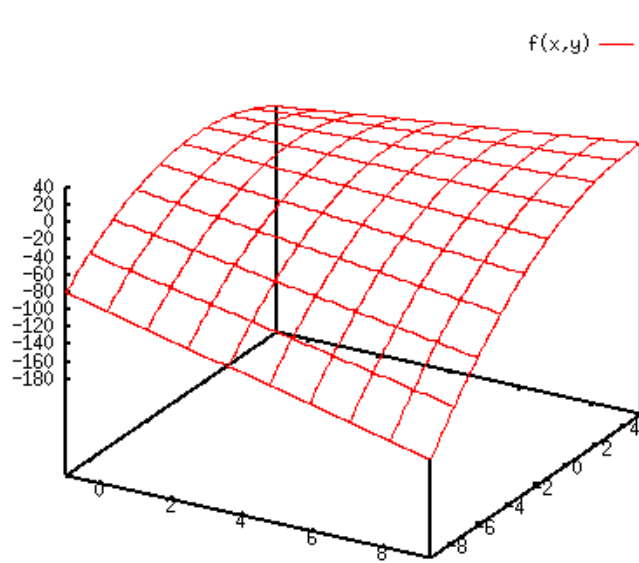
splot 命令可用于绘制 3d 图形。lines 样式将使用方格网络表示平面。pm3d 样式使用实心平面。

$$f(x,y)=x*y-y**2$$

左图 splot f(x,y) with lines

右图 splot f(x,y) with pm3d





3d to 2d 映射(set view map)

spplot 命令在使用 set view map 命令后将垂直映射图形到 xy 平面。

第三部分

命令

本章讲述 gnuplot 内置命令，他们按照字幕顺序排列。事实上某些系统上可能有些命令没有列在此处。

当缩写无歧义时可以省略字母。如：`p f(x) w li` 是 `plot f(x) with lines` 的简写。本文档的语法描述开花括号表示可选参数，管道符号表示多选一。

cd

`cd` 命令更改当前工作路径。

```
cd 'string'
```

路径必须以字符串格式传入，这意味着他们必须被引号括住。建议 windows 用户使用单引号，因为 win 路径中的反斜线在双引号中有特殊意义，他们会被转义。

```
cd "c:\newdata"
```

将会失败。

但是：

```
cd 'c:\newdata'  
cd "c:\\newdata"
```

按照期望工作。

call

`call` 命令用于从文件中载入命令，你最多可以使用 10 个参数。他们在文件中的行被读取后进行替换。当行从文件中读取，系统扫描 `$n` 格式的文本，`n` 介于 0-9。随后他们被替换成 `call` 命令中对应的参数。如果参数在 `call` 命令中利用字符串传入，它将会被替换成无引号字符串。`$#` 等于传入参数数目。`$$` 等于 `$` 本身。

语法：

```
call 'filename' <parameter-0> <parm-1> <parm-2> ...<parm-9>
```

文件名必须以字符串类型出现。建议参数也用字符串格式传入，未来版本的 gnuplot 可能区别对待引用和未引用参数。

例如：

文件 `'calltest.gp'` 包含命令：


```
print "argc=$# p0=$0 p1=$1 p2=$2 p3=$3 p4=$4 p5=$5 p6=$6 p7=x$7x"
```

键入命令

```
call 'calltest.gp' "abcd" 1.2 + "'quoted'" -- "$2"
```

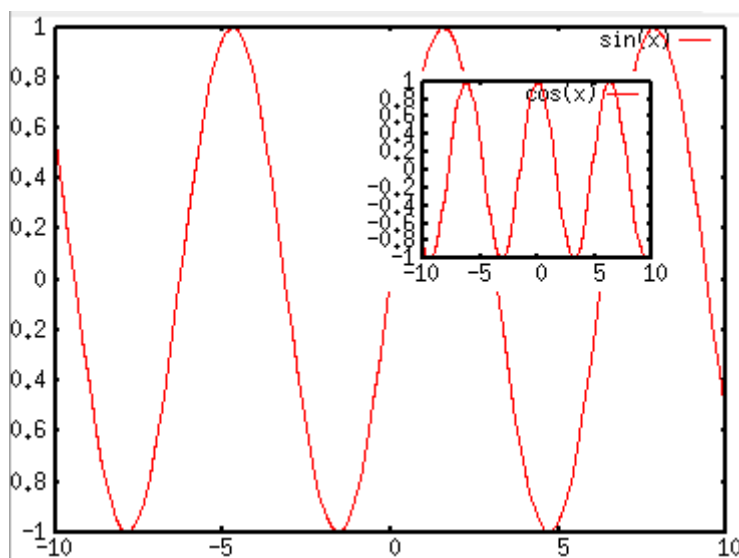
输出：argc=7 p0=abcd p1=1.2 p2=+ p3='quoted' p4=- p5=- p6=\$2 p7=xx

clear

clear 命令清空当前窗体，或者清空 **set output** 命令指定的文件。这通常用于让打印设备进纸。对于某些终端 **clear** 可以只擦除由 **set size** 定义了整个画布的一部分。它可结合 **set multiplot** 命令创建内嵌图形。

例子：

```
set multiplot
plot sin(x)
set origin 0.5,0.5
set size 0.4,0.4
clear
plot cos(x)
unset multiplot
reset
```



do

语法：

```
do for <增量表达式> {
    commands;
```

```
}
```

循环原执行一个代码块。类似 bash 中的 for in。命令必须被花括号括住，并且左花括号必须与 do 在同一行。这个命令不能与旧版本的无括号的 if else 语句合用。

例子：

```
set multiplot layout 2,2
do for [name in "A B C D"] {
  filename=name . ".dat"
  set title sprintf("Condition %s",name)
  plot filename title name
}
unset multiplot
```

evaluate

evaluate 命令执行字符串。换行符不可在 eval 参数内。这对于执行相似格式的命令非常有用：

```
set_label(x,y,text)=sprintf("set label '$s' at %f,%f point pt 5",text,x,y)
eval set_label(1. , 1., 'one/one')
eval set_label(2. ,1. , 'two/one')
```

参阅：substitution macros。

exit

exit 和 quit 命令，类似文件结束符 EOF，结束当前输入流（终端会话，管道，文件读取）如果输入流是局部的（由 load 命令产生），那么父进程依然运行。

exit gnuplot 命令不论当前程序流是否是局部的，无条件退出整个 gnuplot 程序。这种情况下输出文件可能不完整。

```
bind "ctrl-x" "unset output ; exit gnuplot"
```

fit

fit 命令进行曲线拟合。它利用用户函数拟合一组数据。使用非线性最小二乘法（NLLS）Marquardt-Levenberg 算法。在表达式中可以出现任意的用户变量，拟合变量也在其中。表达式的返回值必须是实数。语法：

```
fit {<ranges>} <exp>
'datafile' {datafile-modifiers}
via 'param file' | <var1>{,<var2>,<var3>...}
```

ranges 范围定义了临时限制，当数据点超出这个界限，将会忽略。ranges 语法为：

```
{dummy_variable=} {min} {:max}
```

exp 是任意有效的 gnuplot 表达式，它可以使用用户定义函数 f(x) or f(x,y)。

datafile 和 plot 命令一样。plot 命令所有的修饰符 (using、every、smooth) , 都可用于 fit 命令。

拟合函数的默认数据格式为 z 或者 x:z. 当只有一个输入列时, x 为对应行号。如果有两列,

数据格式可以利用 using 修饰符, 例如希望从一个特殊列中读取 z 的值, 或者从表达式计算 z。一个三列 using 修饰符被应用, 它将被解释成 z 的标准差。标准差用于计算点的权重。如果你没有指定 using 选项, 不从文件中读取 z 标准差, 甚至文件只有 3 列数据。

拟合一个 2 个自变量的函数 $f(x,y)$, using 需要 4 个输入列, x:y:z:s。所有行都必须有这 4 个输入列, 如果字段没有默认值, 这意味着如果字段缺少了, 拟合无法完成。每个数据点的权重根据上一个标准差计算。如果误差估算不能应用, 一个常数可以作为一个常量表达式被应用。

拟合函数最多拥有 5 个自变量。以下允许格式:

z

x:z

x:z:s

x:y:z:s

x:y:t:z:s

x:y:t:u:z:s

x:y:t:u:v:z:s

例如:

```
f(x) = a*x**2 + b*x + c
g(x,y) = a*x**2 + b*y**2 + c*x*y
FIT_LIMIT = 1e-6
fit f(x) 'measured.dat' via 'start.par'
fit f(x) './data/trash.dat' using 1:2:3 via a, b, c
fit g(x,y) 'surface.dat' using 1:2:3:(1) via a, b, c
fit a0 + a1*x/(1 + a2*x/(1 + a3*x)) 'measured.dat' via a0,a1,a2,a3
fit a*x + b*y 'surface.dat' using 1:2:3:(1) via a,b
fit [*:][yaks=*:] a*x+b*yaks 'surface.dat' u 1:2:3:(1) via a,b
fit a*x + b*y + c*t 'foo.dat' using 1:2:3:4:(1) via a,b,c
h(x,y,t,u,v) = a*x + b*y + c*t + d*u + e*v
fit h(x,y,t,u,v) 'foo.dat' using 1:2:3:4:5:6:(1) via a,b,c,d,e
```

详情查阅源代码下的 demo/fit.dem 拟合实例。

每次迭代之后, 当前拟合状态被显示到标准输出, 同时初始和结束状态被写到日志文件, "fit.log".新的内容总是被添加到日志文件中, 无须担心会丢失以前的信息。使用 set fit logfile 修改日志文件名。

当 gnuplot 以合适的选项编译, 使用 set fit errorvariables, 保存拟合中参数的误差值到变量中, 误差变量名是原变量后缀 "_err"产生。这些值可用于更多的计算。

ctrl-c 用于打断当前拟合操作。在当前迭代完成后，1) 可以使用选项去终止拟合或者判断变量值，2) 也可以继续迭代，3) 执行环境变量 FIT_SCRIPT 中的 gnuplot 命令，默认 FIT_SCRIPT 的值为 replot。

一旦 fit 执行完毕，update 命令可用于将最终参数值保存进文件，这个文件可和函数一同加载。

待校准参数

via 关键字有两种方法指明待校准参数，直接或间接在命令行中指定参数表，或者从文件中加载。第二种方法通常用于有初始值的计算。

via 关键字后使用逗号分割的待校准变量列表。如果待校准变量尚未定义，默认初始值为 1.0。无论如何，当有合适初始值时，拟合过程能够迅速完成。

在参数文件中每个参数都需指定个初始值，每行一个：

```
varname=value #注释
```

注释使用# FIXED,有特殊含义。

```
varname=value # FIXED
```

gnuplot 将以 value 为初始值，但是 **fit** 并不校准这个变量。这对于指明参数值很有用。# FIXED 关键字必须严格遵循此格式，否则只是注释。

简要指南

fit 为一组数据根据提供的含有待校准参数的函数计算出待校准参数的值，使得此函数能最大程度和数据重合。拟合的判断基于 SSR (the sum of squared differences or 'residuals')，SSR 利用输入数据和函数在同一横坐标处的值进行计算得到。这个数就称作卡方 (χ^2 希腊字 χ 的平方，统计学中它用于判断观测数据和理论函数的符合程度)。算法试图最小化 SSR，或者尽量精确。

这就是为什么这数学方法叫做最小二乘法 (LLS)。理解非线性的概念，首先需要复习几个概念。在用户函数中使用 z 作为从属变量 (dependent variable，它的值有赖于其它变量)。用户函数可以包含 1 或 2 个自变量 (independent variable，非依赖变量，它独立变化不依赖其它变量)。z=f(x)
z=f(x,y);

系数 (parameter，ax+b a、b 为系数) 是待校准的用户定义变量 (函数定义中未定义变量)。线性和非线性指从属变量 z 关于待校准系数的线性关系，而不是 z 与 x 的关系。对于线性最小二乘，待校准函数是一系列简单函数的组合，不包含任何系数。或者多项式每项只有一个系数。举个例子：

$$z=a*\sin(c*x) + b*\cos(c*x)$$

如果 a b 是待校准变量，c 是常数。那么是一个线性最小二乘问题。如果 c 也是待校准变量，那么就是非线性最小二乘。线性情况下，系数值可以通过简单线性代数方法决定。

fit 命令在拟合中选择最小值。利用一次校准迭代出的系数值计算 WSSR，然后 ML 算法选择下一次迭代使用的系数值，一直循环直到满足预定义的标准。此时就称作曲线完成拟合。迭代 WSSR 的改变值小于 FIT_LIMIT，迭代次数最多为 FIT_MAXITER。用户变量 FIT_CONVERGED 包含 1 或 0，当拟合成功完成为 1，否则为 0。通常函数拟合基于描述数据走势的模式。它用于判断函数符合模式的成度，并且计算系数的校准误差。查阅：fit error_estimates。

如果你希望穿越数据的曲线是平滑的在 fit 指令使用 smooth 关键字。

误差估算

fit 命令中 术语"误差"用在两种上下文中，数据误差估算、系数误差估算。在计算"剩余平方和"权重、WSSR、卡方值时数据误差估算用于计算数据的相对权重。它将影响系数校准，因为它们决定了数据点在函数最终形式上的重要性。包括系数误差估算。当支持精确数据误差计算是非常有意义的。

控制 fit 命令

fit 命令行为受到一些 gnuplot 变量的影响。

控制变量：

FIT_LIMIT

相邻两次迭代的剩余平方和改变小于此变量，那么此时曲线被认为已经拟合。

FIT_MAXITER

控制拟合算法中迭代次数最大值。值为 0 表示无限制。

如果你足够了解 Marquardt-Levenberg 算法，并且希望深度控制拟合算法。gnuplot 提供了几个变量。lambda 初始值根据 ML 矩阵自动算出。利用 FIT_START_LAMBDA 变量手动指定一个值。如果设置此变量值 ≤ 0 ，系统将再次自动计算此变量值。FIT_LAMBDA_FACTOR 变量设置 lambda 增长或减少的因子。

FIT_SKIP FIT_INDEX 变量作为 fit 命令补丁:gnufit，曾用于 gnuplot 早期版本，现在已废除。对数据文件进行多分支拟合时使用 every 关键字，可完成 FIT_SKIP FIT_INDEX 变量的功能。

环境变量

环境变量在 gnuplot 启动前在 shell 中定义。具体如何设置环境变量，因操作系统不同而不同。

FIT_LOG

设置拟合命令日志文件的文件名和相对或绝对路径。默认值为 工作路径下"fit.log"。gnuplot 运行时使用 set fit logfile 修改此设置。

FIT_SCRIPT

设置一个 `gnuplot` 命令，它在 `fit` 过程被用户打断后自动执行，默认值为 `replot`。不过 `plot` 或者 `load` 指令也很有用。

多分支

多分支拟合，多组数据可以用于部分相同待校准系数的不同函数同时拟合。不同的函数和待校准系数（分支）利用伪变量区分分支，例如 虚拟列：数据集记录索引 0 数据块索引-1 或数据集索引-2 作为第二个主变量。

例：已知两个指数衰减函数，他们格式上都符合 $z=f(x)$ 。分别描述不同数据，但是他们拥有相同衰减时间。假设数据文件格式为 `x:z:s` 那么：

```
f(x,y)=(y==0)? a*exp(-x/tau) : b*exp(-x/tau)
fit f(x,y) 'data' using 1:-2:2:3 via a,b,tau
```

其它复杂实例参看源代码目录下：`fit.dem hexa.fnc`

help

`help` 命令显示内建帮助。指定帮助主题显示对应文档：

```
help <topic>
```

未指定帮助主题，将出现主题列表和其它信息供用户选择。当 `help` 参数只有问号时，只提供主题列表。

history

`history` 命令列出、保存或执行曾经输入的命令。简要 `usage`：

```
history                #显示全部历史记录
history 5              #显示最近输入的5行命令
history quiet 5        #显示最近5行历史，但不包含项目序号。
history "hist.gp"      #保存全部历史记录到文件 hist.gp，文件原内容丢失。
history "hist.gp" append #添加模式保存全部历史记录到 hist.gp
history 10 "hist.gp"   #保存最近10条记录到文件 hist.gp
history 10 "|head -5 >>diary.gp" #写入文件到管道。
history ?load          #显示所有以 load 开始的命令
history ?"set c"       #显示所有以 set c 开始的命令
history !reread        #执行以 reread 开始的最近一条命令。
history !"set xr"      #类似上面。
```

if

新语法

```
if ( 条件 ){cmd; cmd;  
cmd;  
cmd;  
} else {  
cmds;  
}
```

旧语法：

```
if (条件) 命令行 [; else if ( 条件 ) ... ; else ...]
```

如果 if else 关键字后紧跟 左花括号那么此 if 使用新语法，否则使用旧语法。意味着左花括号必须与 if else 在同一行。旧语法依旧支持，但不能混合使用花括号和无花括号格式。

旧 if

4.4 版本 gnuplot 将 if else 命令限制在一行上。但现在可以使用花括号以多行支持。旧语法在新版本中依旧有效但它不能用在花括号内部。

例：

```
pi=3  
if (pi!=acos(-1)) print "?Fixing pi!"; pi=acos(-1); print pi
```

将显示：

?Fixing pi!

3.14159265358979

但是

```
if (1==2) print "Never see this"; print "Or this either"
```

无任何显示

除非：

```
v=0  
v=v+1; if (v%2) print "2" ; else if (v%3) print "3"; else print "fred"
```

迭代

do plot splot set unset 命令可包含 for 迭代子句。它将同一命令执行多次。每次根据控制变量重新计算表达式值。do 命令用于执行任意命令。

for 子句为如下格式：

```
for [intvar = start:end{:increment}]  
for [str in "A B C D"]
```

例如：

```
plot for [filename in "a.dat b.dat c.dat" ] filename using 1:2 with lines  
plot fir [basename in "a b c"] basename . ".dat" using 1:2 with lines  
set for [i=1:10] style line i lc rgb "blue"  
unset for [tag = 100:200] label tag
```

嵌套的 for 子句也被支持：

```
set for [i=1:9 ] for [j=1:9] label i*10+j sprintf("%d",i*10+j) at i ,j
```

do 命令用于执行任意命令：

```
do for [i=1:9] {  
    print i;  
}
```

load

load 命令读取指定文件，并且执行它们。文件通过 save 命令创建，load 命令加载。任何包含正确命令的文本文件都可被 load 加载。

```
load "input-file"
```

文件名必须包含在引号内。特殊文件名 "-", 表示从标准输入读取命令。一些 OS 支持管道操作：

```
load "< generator.sh"
```

此命令利用管道从 shell 脚本读取输出。注意此时的操作符是 < 而不是 |, save 命令使用 | 表示管道输出。

pause

pause 命令 显示文本并且暂停。语法：

```
pause <time> {"text"}      #显示 text 暂停指定时间。  
pause mouse { <endcondition> } {, <endcondition> } { "string" }
```

time 表示暂停的时间。值为 -1 等待直到有键盘操作，为 0 不等待，为正数等待指定秒。pause 0 类似与 print 语句。如果当前终端支持鼠标，pause mouse 等待鼠标单击或者 ctrl-c。对于其它终端 pause mouse 等价于 pause -1.详情见：鼠标输入

如果多个结束条件在命令中给出，那么这些事件可以结束暂停过程。可用的结束条件为：keypress, button1, button2, button3, close, any。如果暂停被键盘打断那么键盘键值保存在 MOUSE_KEY, 键字符串格式保存在：MOUSE_CHAR。鼠标坐标可通过鼠标变量：MOUSE_X MOUSE_Y MOUSE_X2 MOUSE_Y2，详情查阅：鼠标变量


```

pause -1 # 等待直到键盘操作。
pause 3 #等待3秒
pause -1 "Hit return to continue" #同 pause -1,同时显示一字串。
pause 10 "Isn' t this pretty? It' s a cubic spline."
pause mouse "Click any mouse button on selected data point"
pause mouse keypress "Type a letter from A-F in the active window"
pause mouse button1,keypress
pause mouse any "Any key or button will terminate"

```

如果你希望等待某个特殊键，需要使用 reread 循环：

```

print "press tab in the plot window"
load "wait_for_tab"

```

文件 wait_for_tab 内容为：

```

pause mouse key
if (MOUSE_KEY != 9) reread

```

Plot

plot 是主要的绘制命令。他提供多种方式根据函数或者数据绘图。plot 用于 2d，splot 用于 3d。plot 和 splot 有很多性质是相同的。特别的：binary 关键字在 plot 和 splot 下都能工作，但是有少许不同。

语法：

```

plot {<ranges>}
    {<iteration>}
    {<function> | {"<datafile>" {datafile-modifiers}}}}
    {axes <axes>} {<title-spec>} {with <style>}
    {, {definitions{,}} <function> ...}

```

<function> 函数 和被引号护住的文件名 <datafile> 都可用于绘图。一个函数是一个数学表达式。函数可以是 gnuplot 内建的，也可以是用户定义的，或者是 plot 命令内建的。使用一个 plot 命令可以绘制多个函数，它们之间使用逗号分隔。

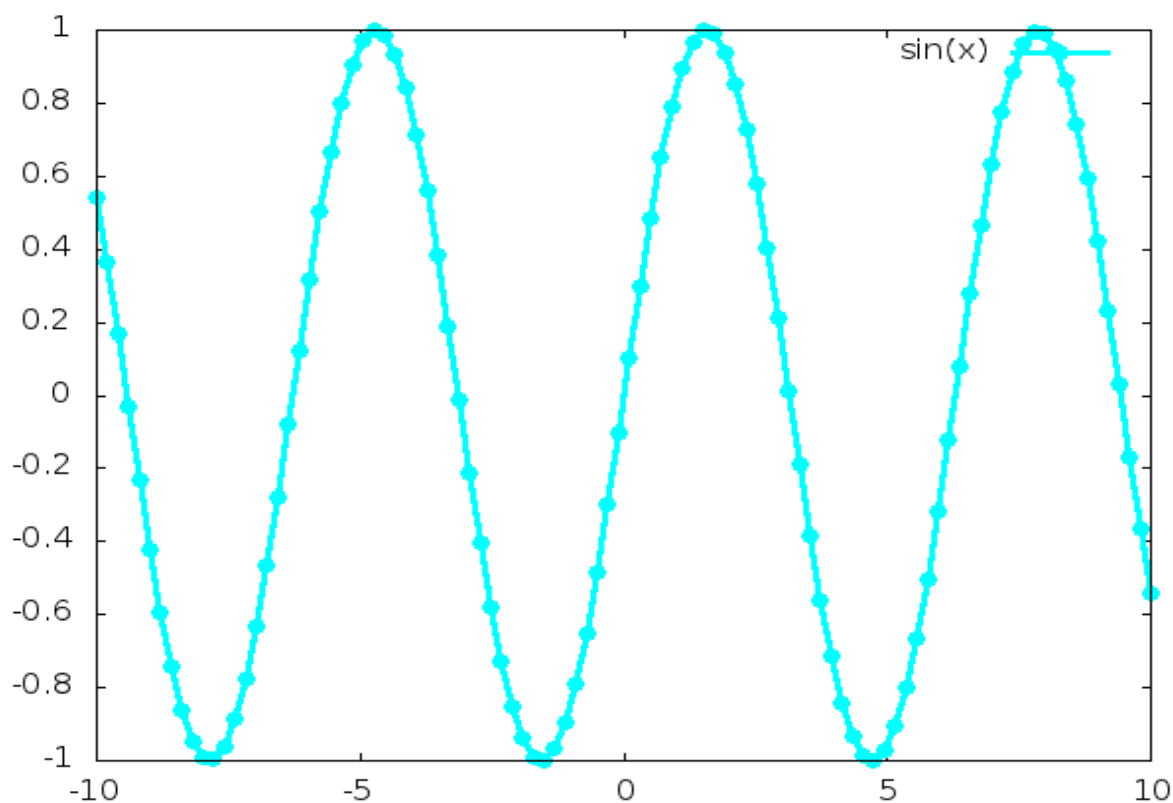
```

plot sin(x)
plot sin(x), cos(x)
plot f(x) = sin(x*a), a = .2, f(x), a = .4, f(x)
plot "datafile.1" with lines, "datafile.2" with points
    [t=1:10] [-pi:pi*2] tan(t), \
    "data.1" using (tan($2)):(($3/$4) smooth csplines \
    axes xly2 notitle with lines 5
plot for [datafile in "spinach.dat broccoli.dat"] datafile

```

关键字 : axes

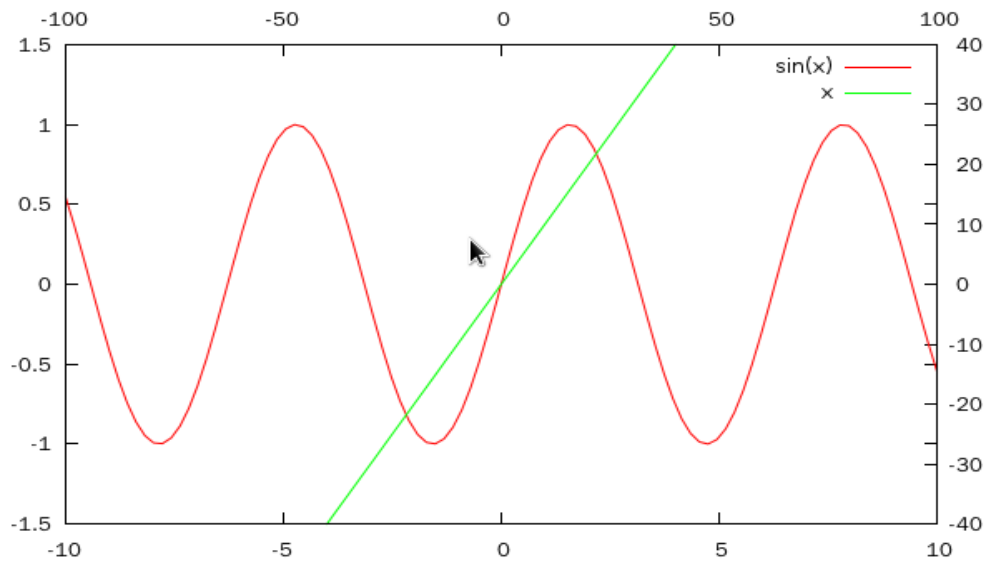
gnuplot plot 在图片上放置了 4 个坐标轴，2 个横轴 2 个纵轴，合计可以拥有 2*2 个坐标系。一次绘制可以使用 axes 命令指定某一坐标系。



如上图，横坐标有两个 底部一个 顶部一个分别命名为 x1 x2。纵坐标有两个 左边一个 右边一个 分别命名为 y1 y2。四个坐标系分别为：x1y1 x2y2 x1y2 x2y1。

例子：

```
set yrange [-1.5:1.5] #设置范围
set x2range [-100:100]
set y2range [-40:4]
set xtics;set x2tics;set ytics ;set y2tics;#显示坐标轴刻度。
plot sin(x) axis x1y1,x axis x2y2;#在一副图上绘制两个图形分别使用不同坐标系。
```



自行观察图形四周的坐标轴和代码的关系。set xlabel 可以设置坐标轴的名字。

关键字：*binary*

二进制文件：

旧版的 gnuplot 可以自动识别二进制文件。不过对于新版 gnuplot 需要在文件名后加上 `binary` 关键字。

处理二进制文件需要在命令中提供足够的文件结构信息，或者使用 `filetype` 关键字自动解析数据文件。二进制文件有两种结构。二进制矩阵格式（`binary matrix format`）和 通用二进制格式（`binary general format`）

矩阵格式包括一个的 2 维矩阵，元素类型为标准 IEEE 定义的 32bit 浮点类型。在 `using` 关键字中，第一个输入列代表数组元素所在行，第二个输入列代表数组元素所在列，第三个输入列代表元素值。

通用二进制格式可以包含任意数目的列，在命令行中需要指定更多的信息。例如：`array record format` 和 `using` 关键字分别表示了 大小，格式，维度。有一些命令可以从文件中跳过一部分字节，读取文件的一部分。通用二进制对待 第 1 2 3 输入列和 矩阵格式不同。

有一些全局设置作为 plot 的默认设置，它们使用 `plot <filename> binary....` 相同的语法：`set datafile binary`。在命令中指定的参数将覆盖默认设置。

如果 `binary` 没有带上 `general` 关键字，那么 `matrix` 作为默认值。

如果 `plot` 命令 的文件名 使用一个连字符 - 那么自动进入通用二进制模式，并且从管道中读取数据。此功能允许 gnuplot 同其他程序进行数据交换。此处没有 数据结束标记。gnuplot 将读取 `array` 指定的数据长度后停止。

`index` 关键字不被支持，`using` 和 `every` 关键字可用。

参考：Binary File Splot Demo. 实例。

通用二进制格式文件中可以不包含结构信息。这样的话就需要在命令行中指定这些信息。虽然语法比较晦涩，但是此格式对 gnuplot 与其他软件交互提供了好处。

语法：

```
plot ' <file_name>' {binary <binary list>} ...  
splot ' <file_name>' {binary <binary list>} ...
```

通用二进制格式由<binary list>内的关键字激活，i.e. array record format filetype。其他情况默认使用矩阵格式。

关键字 filetype 影响文件读取。想知道哪些文件类型被支持，运行：show datafile binary filetypes。

通用二进制格式可总结为 2 个基本类型。类型取决于数据的处理方法。第一个是均匀采样，并且点的坐标也必须生成。这一类型可以被 <binary list> 关键字完整控制。另一类型非均匀采样，他是一系列文件的集合，点的坐标信息在文件里。

其他的罕见文件，一个应该是 ASCII 数据。

关键字：array

设置二进制数据的维度宽和高。行与列坐标由 gnuplot 自动产生。每个坐标指定一个整数，例如：array=(10, 20)。它意味着一行有 10 个元素，然后延伸 20 行。负数表示数据一直被读取，直到文件结束。如果只有一维那么，括号可以省略。另外冒号可以用于分割不同数据。Array=25:35，表示有两个一维数组，第一个包含 25 个元素第二个包含 35 个元素。

注意 gnuplot 4.2 使用 array=10X20，旧语法已经抛弃，不过 gnuplot 可以按照兼容方式编译，

关键字：record

此关键字提供和 array 一样的功能，不同点在于，record 不会自动产生坐标。当坐标信息保存在数据文件中的某列时，你需要此选项。

关键字：skip

假设数据文件有 1024byte 的文件头，使用 skip 关键字可以跳过它。如果不同 array 之间存在间隔也使用 skip 跳过。例如：plot 'file.data' binary record=356:356:356 skip=512:256:356

此命令让 gnuplot 跳过 512byte 后读取第一个 record（356 字节），再跳过 256byte 后读取第二个 record，再跳过 356byte 后读取第三个 record

关键字：format

默认二进制元素类型是 float（浮点类型）。使用 format 选项用来指定不同的数据类型。例如：

format="%uchar%int%float" 代表数据文件二进制 array 结构，第一字段为 uchar 类型，第二字段为 int 类型，第三字段为 float 类型。如果 format 的元素个数少于总字段个数，那么其他的字段将默认使用 format 最后一个类型。使用 星号* 可以指定抛弃的字段,使用整数可以自动重复。如：format="%2int%*2uchar %3float"，它告诉 gnuplot 读取 2 个 int 类型，然后跳过 2 个 uchar 数据后读取 3 个 float 类型。

使用 show datafile binary datasizes 查看可用类型，包括硬件依赖的类型和硬件不依赖的类型。

关键字：endian

处理二进制数据需要处理端序。因为二进制数据并不总是来自于 gnuplot 运行的计算机。endian=little,指定 gnuplot 使用小端序。Endian 可用的选项如下：

little: 小端端序。

big: 大端端序。

default:使用与编译 gnuplot 编译器相同的端序方案。

awap (swab):interchange the significance。

gnuplot 支持中端序 (middle)，gnuplot 需要使用特殊选项编译。

关键字：filetype

某些标准二进制格式，gnuplot 可以从文件中得到必要的信息去解析数据。例如：format=edf，将读取 edf 的文件头。使用 show data file binary filetype，显示可用的类型。

format=auto，将使得 gnuplot 通过文件扩展名自动选择文件类型。

命令行选项可以覆盖标准文件类型的读取方式，查看 set datafile binary 获取更多信息。

avs 格式是自动识别的图片二进制格式。avs 是非常简单的格式，非常适合应用程序之间的数据交换。它由 2 个 long 类型数据开始 (xwidth , ywidth)，后面跟上二进制流。每个数据包包含 4byte (alpha red green blue)。

edf 格式是自动识别的图片二进制格式。edf 代表 ESRF data format，它支持 edf 和 ehf (ESRF header format) 格式。

<http://www.edfplus.info/specs>

获取更多信息。

Png 如果 gnuplot 的编译使用了 libgd 库去支持 png/gif/jped 输出，那么这些文件格式也可以被读入。你可以明确指定读取格式：plot 'file.png' binary filetype=png 或者使用：plot 'file.png' binary filetype=auto，自动识别 png 格式。

其他关键字

此处列出的关键字只在自动产生元素坐标时起控制作用。即，控制坐标(x,y,z)与元素的对应关系。

Scan 更改数组坐标在坐标系中的对应关系。plot 中 scan 可以使用 xy 两个字母。Splot 中 scan 可以使用 xyz 三个字母。

gnuplot 读取数据内部处理为数组，数组元素的索引从 0 开始。

gnuplot 'dat.dat' binary array=(3,6) using sprintf("%d", \$3) with labels

观察上述命令,你可能发现 label 样式需要 3 个输入列 (x:y:text)，但是 using 只指定了一项(text)。因为 array 关键字自动产生 x y 坐标并且传递给 label 样式。上例会读取 3*6 个 32bit 浮点数据。她们的索引依次为：

(0,0) (0,1) (0,2) |(1,0) (1,1) (1,2)| (2,0)(5,2)。

在 array 关键字下，配合 scan=xy 将导致元素的绘制坐标 (x,y)=索引。元素(0,1)，绘制在 0,1 坐标处，元素(x,y)绘制在(x,y)坐标处。

scan=yx 将导致(y,x)=索引。也就是交换了横纵坐标。元素(0,1)，绘制在 1,0 坐标处，元素(x,y)绘制在(y,x)坐标处。

Splot 情况与此类似。如果坐标系不是笛卡儿坐标系，而是极坐标系，那么 xyz 依次换成 t r z。

dx,dy,dz 当 gnuplot 产生坐标时，控制点在各个坐标轴上的间距。以下讨论基于。

scan=xy dx=10 dy=5, 元素(0,0)绘制于坐标(0,0)，元素(x,y)绘制于坐标(x*dx,y*dy)。

scan=yx dx=10 dy=5, 元素(0,0)绘制于坐标(0,0)，元素(x,y)绘制于坐标(y*dx,x*dy)

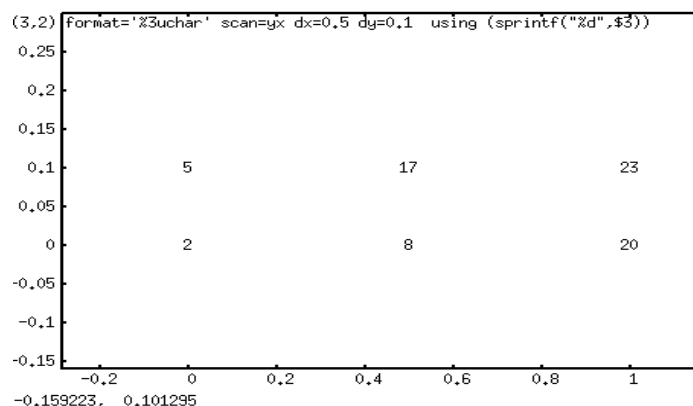
实例：

shell:

```
echo '$\x0\x1\x2\x3\x4\x5\x6\x7\x8\x9\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x20\x21\x22\x23'
>binary.dat
```

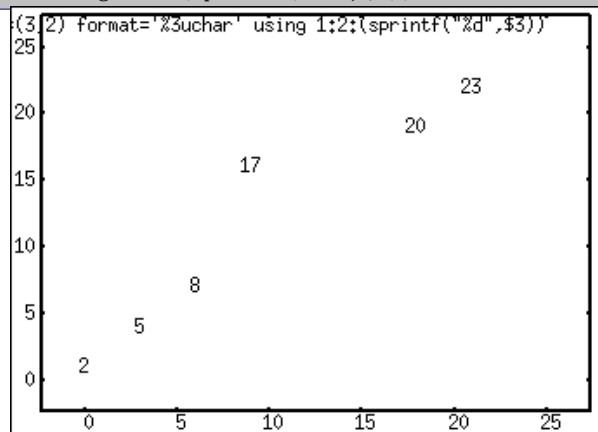
gnuplot:

```
plot 'binary.dat' binary array=(3,2) format='%3uchar' scan=yx dx=0.5 dy=0.1 \
using (sprintf("%d", $3)) with labels
```



输出六个数值，因为读取了 6 个元素。

```
plot 'binary.dat' binary record=(3,2) format='%3uchar' scan=yx dx=0.5 dy=0.1 \
using 1:2:(sprintf("%d", $3)) with labels
```



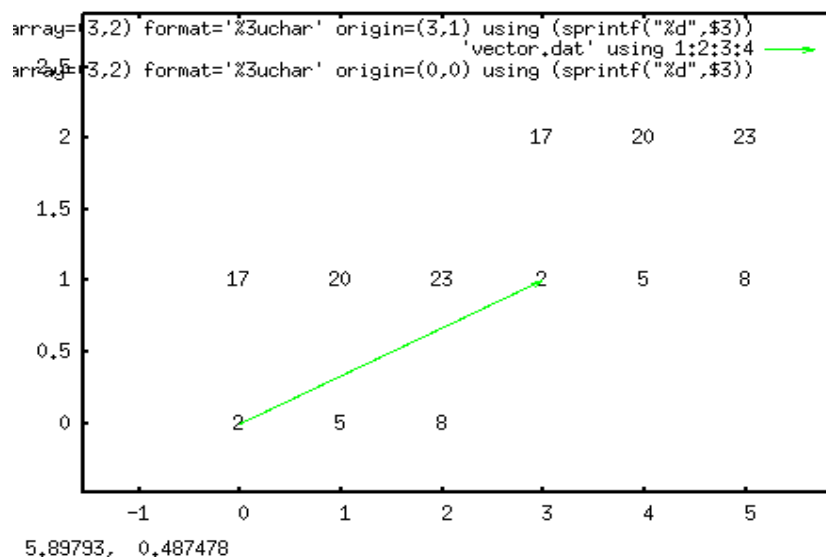
使用 record 关键字代替 array 关键字，由于 record 不自动传递坐标给样式，所以 using 需要写出这两个输入列。这意味着绘制将使用输入字段 1 和字段 2 的值来设置标签坐标。

tranpose 矩阵转置，等价与：scan=yx or scan=yzx。

flipx flipy flipz 某些情况，扫描文件方向和你需要的并不一致。使用 flip 在某一维度上逆序。

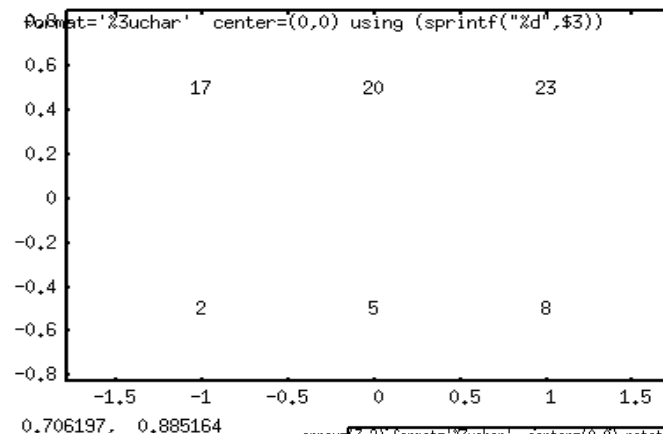
Origin 在转置和 flip 运算后。array 自动产生的坐标，元素(0,0)总是绘制在 (0,0)，并且 x y 依次递增。如果你有多个数据，使用 origin 选项控制(0,0)号元素绘制起点，相当于图像平移。

下图是上例分别添加 origin=(0,0) 和 origin=(3,1) 绘制。



如果你有多个数据可以使用 origin=(0,0):(0,100) 设置每个 array 的绘制方法。在 splot 命令中 origin 可以使用 3 个坐标。

center 此关键字，类似 origin 关键字，将图像中心绘制到指定坐标，center 不能与 origin 关键字一同使用，因为它们都是决定整个图像的位置。上例使用 center=(0,0)的效果：

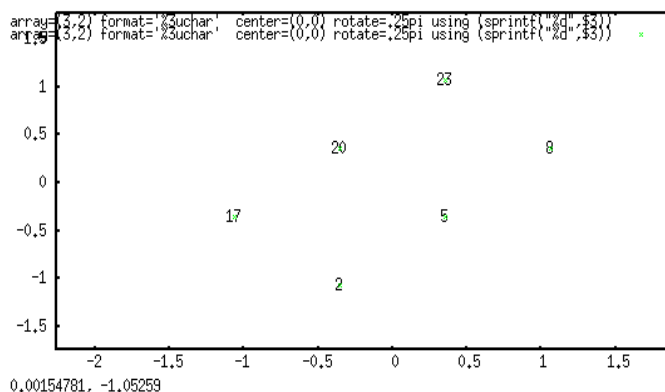


虽然没有设置 origin 选项，但是整个图像起点被设置到了(-1,-0.5xx) 使得整个图形中心在坐标(0,0)。

rotate 为了提供更大的自由，可以指定角度旋转图形。rotate=45deg or rotate=0.25pi.

右图是使用了 rotate=.25pi 的图形：

可以看到整个图形被旋转了，由于横坐标轴和纵坐标轴的显示比例不同所以图形不是矩形。



查阅 set xrange , set yrange ,set xtics , set ytics

perpendicular 对于 splot 指定一个透视变换。默认：perp= (0,0,1)。

The two-dimensional rotation is done first, followed by the three-dimensional rotation. That is, if R' is the rotational 2×2 matrix described by an angle, and P is the 3×3 matrix projecting $(0,0,1)$ to (x_p, y_p, z_p) , let R be constructed from R' at the upper left sub-matrix, 1 at element 3,3 and zeros elsewhere. Then the matrix formula for translating data is $v' = P R v$, where v is the 3×1 vector of data extracted from the data file. In cases where the data of the file is inherently not three-dimensional, logical rules are used to place the data in three-space. (E.g., usually setting the z-dimension value to zero and placing 2D data in the x-y plane.)

数据

一个文件中可以包含分散的数据。

语法：

```
plot ' <file_name>' {binary <binary list>}
  {{nonuniform} matrix}
  {index <index list> | index "<name>"}
  {every <every list>}
  {thru <thru expression>}
  {using <using list>}
  {smooth <option>}
  {volatile} {noautoscale}
```


关键字，binary index every thru using smooth 将分开讨论。Binary 前文已述。

纯文本数据 (ASCII DATA FILE)

数据文件一条记录¹代指一条数据。using 可以指定引用字段。gnuplot 自动忽略以 # 开始的行。可以用作数据的注释。一个文件中的多条记录之间应该由至少一个空白行分隔，注释行不算。字段由空白分割，单引号和双引号可用作字符串数据。

```
1.0      "second column"      3.0e3
```

此行包括 3 个字段。第一个字段为 1.0 第二个字段为字符串，第三个字段为 3.0×10^3 。科学记数法使用 e 或者 E 分割。如果设置了 set datafile fortran, fortran 指数格式 d D q Q，也可以使用。

自动缩放 set autoscale，被开启那么图形将自动缩放将所有点显示在输出，如果开启 set xtic，那么会给大量的 tics 上标记。这会有两个结果：1) 对于 splot，面的转折可能与原先不匹配。这种情况下，不会绘制任何竖直线，2) 利用同样的横坐标在 x1 和 x2 上绘制多轴图形，如果 x2tics 没有显示，x 坐标看起来不正确。因为 autoscale 缩放 x 轴，不缩放 x2。为了避免问题，使用 set autoscale fixmin/fixmax，它关闭了 xrange 的自动扩展，转而缩放整个图像。

关键字：every

语法：

```
plot 'file' every {<point_incr>}
               {:{<block_incr>}}
               {:{<start_point>}}
               {:{<start_block>}}
               {:{<end_point>}}
               {:{<end_block>}}}
```

every 指定了数据的采样循环。只有被 every 选择的数据才输出。忽略 every 关键字，导致记录中所有数据都被采样。一个 point 由记录的行决定。一个块是连续数据。块索引和行索引都从 0 开始。

```
Every :::3::3    #采样第四块。
```

```
Every ::::9      #选择前 10 块，等价于：every :::0::9
```

```
every 2:2        #选择绘制 0 2 4 6 8 ... 块，其中的 0 2 4 6 8 ... 行
```

```
every ::5::15    #每个块绘制其中的 5 到 15 行
```

数据文件实例

这个例子使用 Population.dat

Population.dat 包含：

1 一个记录 (record) 代指一行数据。一个文件中可以包含多个数据集 (用连续多个空白行分隔)，一个数据集包含多个数据块 (由 1 个空白行分隔)，一个数据块包含多条记录。

1965 103

1970 55

1975 34

1980 24

1985 10

#####

```
pop(x)=103*exp((1965-x)/10)
set xrange [1960:1990]
plot 'population.dat', pop(x)
```

二进制实例：

前文有例，binary：70。

```
plot 'binary.dat' binary array=(512,1024):(1024,512) format="uchar" \
origin=(0,0):(1024:1024) flipy using 1:2:3 with rgbimage
```

index

Index 关键字可以指定文件内的记录。一个 plot 指令只处理一个记录。如果文件中包含多个集合，需要使用多个 replot 命令分别绘制它们。

语法：

```
plot 'file' index { <m>{:<n>{:<p>}} | "<name>" }
```

数据块由单个空白行分割，数据集由两个以上空白行分割。Index 指定数据集索引（编号从 0 开始）。一次 plot 只处理一个数据集。

Index m 只使用编号为 m 的集合

index m:n 使用编号从 m 到 n 的集合

index m:N:p 选择编号为 $A_n = m + n * p, A_n < N$ 的所有数据。

Index 'name' 用来引用已命名数据起点。数据名称由上方的注释行决定。

假设 file.dat 结构如右图。

Index '==R0=='

将从第 R0 开始处理数据集。由于第二条记录和第一条之间只有一个空行，所以它们同属第 0 个数据集。这意味着 plot 指令将处理 R0 R1 两条记录后停止，不包含 R3。

Index '--R1'

```
#file.dat
#==R0==
1 2 3 4

#--R1
2 2 3 4

#=<R3
3 2 3 4
```

使得 plot 从 R1 记录开始处理，因此 R0 被跳过。

Index ' $\leq R3$ '

使得 plot 从 R3 记录开始。

名称可以缩写： $\leq R3$ 可以写成 $\leq R$ 、 \leq 或 $=$ 。 $\leq R0$ 可以写作 $\leq R$ 、 $=$ 、 $=$ 。

如果写作 $=$, plot 使用第一个匹配的标题 $\leq R0$ ， $\leq R3$ 将不会被使用，

smooth

平滑选项指定如何使数据之间的连线平滑。为了使数据制图美观和合理，根据数据绘制的曲线有不同的平滑算法。曲线拟合 fit 命令也需要使用平滑算法。

语法：

```
smooth {unique | frequency | cumulative | cnormal | kdensity  
       | csplines | acsplines | bezier | sbezier}
```

autoscale 选项打开，将使得平滑曲线总体在图片边界内。

autoscale 没有开启，并且平滑选项为 acspine 和 cspline，曲线采样将穿过输入数据的 x 区域。

Acsplines 自然样条平滑。数据局部关于 x 单调，曲线是分段函数。由系数符合输入权重的，三次多项式构造。权重由第三输入列表示。

```
Plot 'data-file' using 1:2:(1.0) smooth acsplines
```

输入权重的数量级将定性的影响曲线的分段数目。如果某个权重较大，那么它所在点对曲线影响越大。曲线逼近连接各个点的 3 次自然样条曲线。如果权重较小，曲线由数个小片段组成，总体上是平滑的。极限情况是单个片段由加权线性最小二乘法产生，逼近所有数据。数据误差可用作平滑权重。

插图使用的文件

```
13 20  
13 20  
21 10  
10 15  
30 33  
15 17  
31 20  
18 32  
  
未排序文件:  
file.dat
```

```
13 20  
13 20  
21 10  
10 15  
30 33  
15 17  
31 20  
18 32  
  
排序后文件:  
file_sort.dat
```

绘制命令 : plot 'file.dat' using 1:2 <some options>

bezier 贝赛尔选项使用 n 级贝赛尔曲线去逼近数据。

Csplines 在应用 unique , 使用三次自然样条曲线连接连续点。

Sbezier 首先应用 unique , 然后应用 bezier.

unique 此选项使得图形在 x 轴单调¹ (没有两个相同横坐标的点)。如果数据中有两个点拥有相同横坐标那么 , 取其中点。随后各个点之间使用直线端连接。

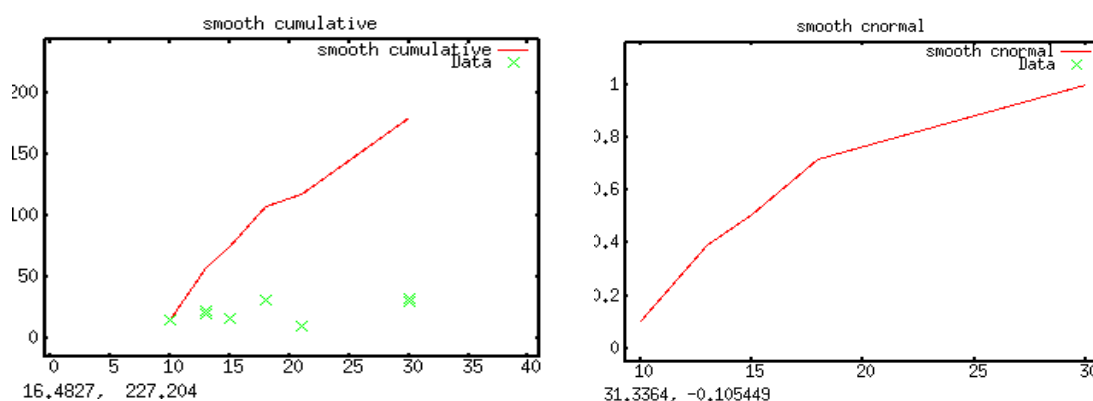
frequency 使得图像在 x 轴单调。拥有相同 x 值的若干点 , 被单个点替换 , 新点 y 值等于若干 y 值的和。(x,y1) (x,y2) 两个点会被替换成 (2,y1+y2)

cumulative 使得图像在 x 轴单调。累积图。当前横坐标的绘制点的 y 值 是当前横坐标数据 y 值和 所有小于当前横坐标的数据 y 值之和。(x,y1) (x,y2) 被替换成(x,(y1+y2)/2)

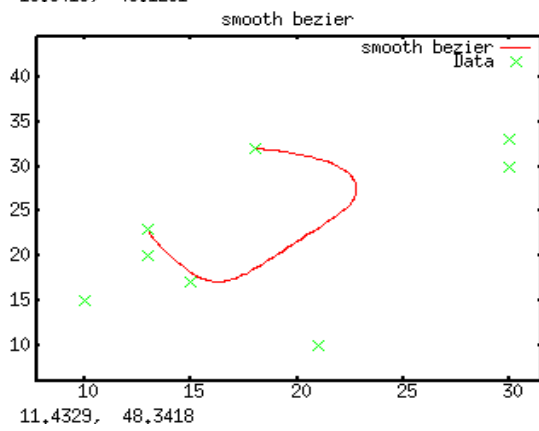
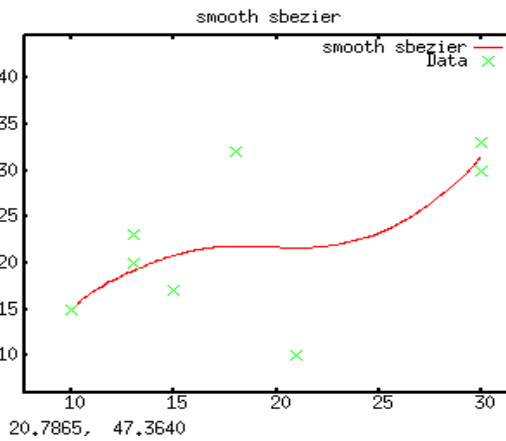
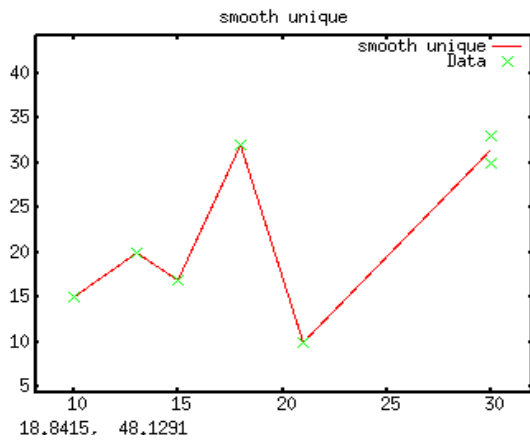
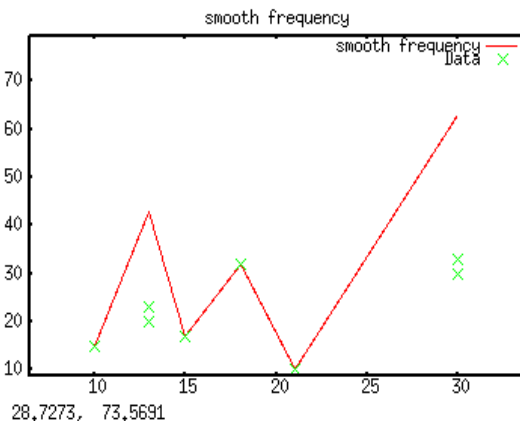
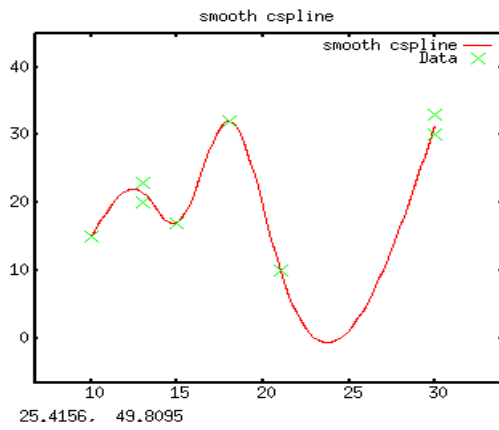
cnormal 使得数据关于 x 单调。把值映射到[0:1]之间。先应用 cumulative 算法得到每个横坐标的累积值 , 当前绘制点为 : 当前横坐标的 cumulative 值 除以 所有数据 y 值之和 , 值域一定在[0:1]内。

kdensity 用于对随机数据绘制核密度估计曲线。The kdensity option is a way to plot a kernel density estimate (which is a smooth histogram) for a random collection of points, using Gaussian kernels. A Gaussian is placed at the location of each point in the first column and the sum of all these Gaussians is plotted as a function. The value in the second column is taken as weight of the Gaussian. (To obtain a normalized histo-gram, this should be 1/number- of-points). The value of the third column, if supplied, is taken as the bandwidth for the kernels. If only two columns have been specified, or if the value of the third column is zero or less, gnuplot calculates the bandwidth which would be optimal if the input data was normally distributed. (This will usually be a very conservative, i.e. broad bandwidth.)

各个平滑选项的效果 :



1 输入数据可能是乱序的, 例如 读取 5 个点 1,2 4,3 2,1 5,2 3,3。图形平滑时不能按照读取时的顺序连接。因为那样就会产生闭合曲线。某些平滑算法是针对函数曲线的, 这意味着对于每个有效的 x f(x)有唯一值。



特殊文件名

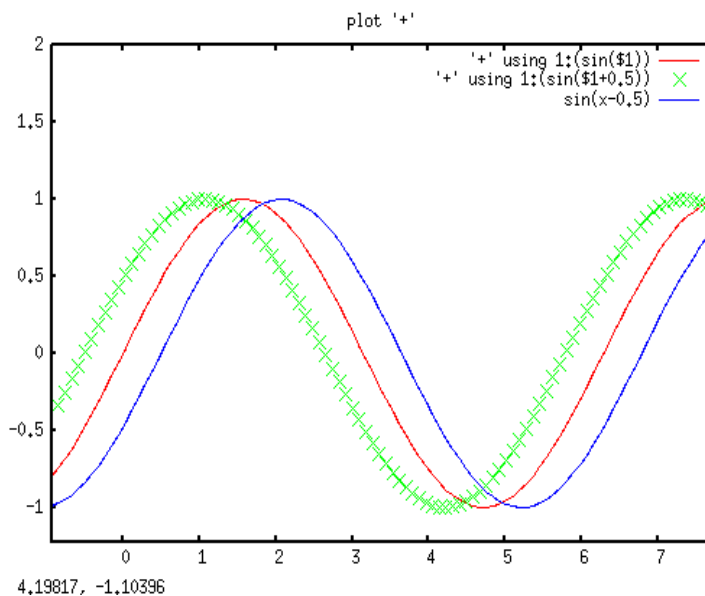
有些文件名有特殊含义，`'-'` `'+'` `'++'`，

空文件名，表示使用最近使用的文件名。`'-'` 表示数据内联在脚本中，紧紧跟着命令之后。`'+' '++'` 提供一种使用内联机制由 `plot` 脚本自动产生数据。

`'+'` 自动产生 1 个自动连续采样的输入数据列，如果 `xrang` 在 `[x0:x1]` 那么只采样 `x0:x1` 区间，`set samples` 设置采样选项：

例如：

```
plot '+' using 1:(sin($1)) with lines
replot '+' using 1:(sin($1+0.5)) with point
replot sin(x-0.5)
```

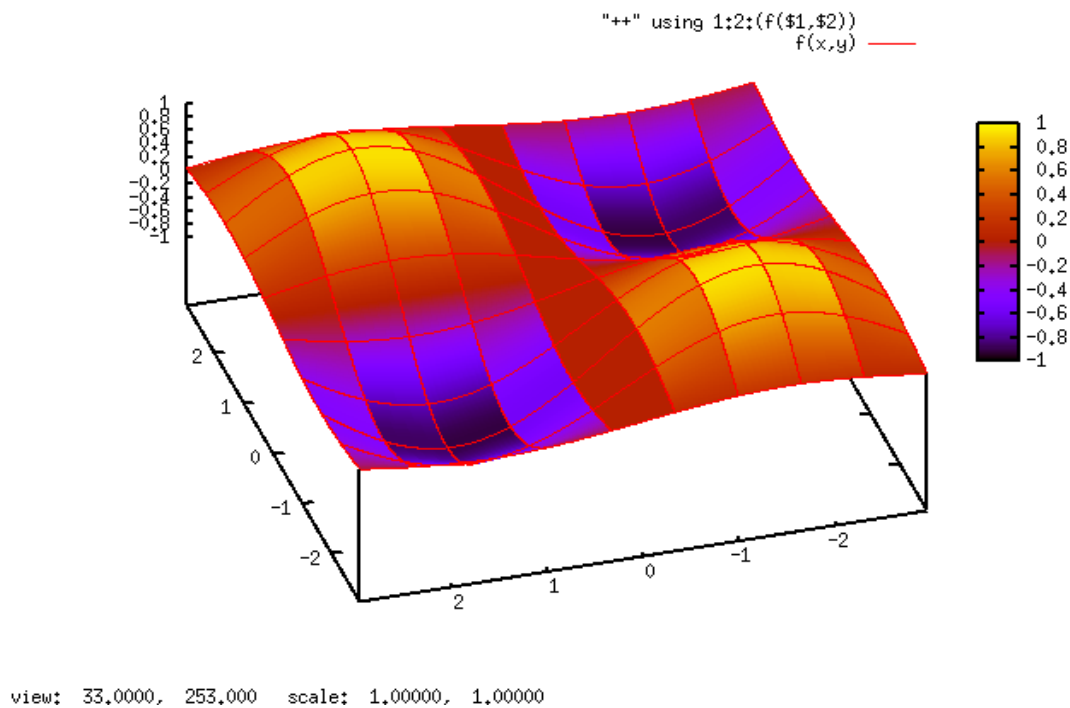


观察上图理解自动连续采样。红色正弦曲线由 `'+'` 绘制。

`'++'` 产生 2 个自动采样输入列。

```
plot '++' using 1:2 with point #产生点阵。
f(x,y)=sin($1)*sin($2)
splot '++' using 1:2:(sin($1)*sin($2)) with pm3d,f(x,y) with lines
```

类似如下绘制：



'-' 文件名，数据紧接着命令之后，如果交互模式下就交互式输入，如果在脚本中数据从脚本中此命令后读取。以e 结束。

```
plot '-' index 0 , '-' index 1
2
4
6
e

10
12
14
e
2
4
6

10
12
14
e
```

在replot 命令中使用 '-' 或者使用" 重复 '-' 文件，那么你需要再次输入数据。

Plot 命令可以使用管道读取其他程序输出.文件名必须以 小于号开始，接上 shell 命令：

```
plot "< sed -n '1,7p' vec.dat" using 2:3 with points
```

使用<管道读取 sed -n '1,7p' vec.dat 命令输出，使用输出的 2:3 列绘制 points

thru 关键字提供老版本兼容

```
plot 'file' thru f(x)
```

等价于

```
plot 'file' using 1:f(x)
```

using

指定传递给绘制样式的参数。

```
Using 3:2:1:4
```

表示 传递数据文件中第 3 字段 第 2 字段 第 1 字段 第 4 字段给绘制样式。3:2:1:4 被称作输入列，第一输入列引用数据文件第 3 字段，第二输入列引用数据文件第 2 字段，第三输入列引用数据文件第 1 字段....。

如果你的数据中 第 3 第 4 字段保存的是 横坐标和纵坐标，使用 using 3:4 with points，将第 3 4 字段传递给 point 样式。字段由 set datafile separator 的值分割，默认为空白。输入列中可以使用内联函数。在输入列需要被圆括号扩住：

```
using 1:($2/$3) with points
```

\$1 \$2 \$3 引用当前行对应字段的值，是 column(1) column(2) 的缩写形式。点横坐标从第一字段，纵坐标等于 第二字段除以第三字段。函数 valid(N) 判断第 N 列是否有效。如果每个数据列都包含一个字段名（label），并且这些字段名在第一行，那么可以使用 column("name")引用对应字段。

¹ Height	Weight	Age
167	120	23
170	135	24
....		

以下命令等价：

```
plot 'data' using (column("Age")):(column("Height"))
plot 'data' using "Age":"Height"
plot 'data' using 3:1
```

使用 set key autotile columnhead 开启此功能。

1 字段名不是以注释出现，所以行首不可出现 #。

空置的输入列默认依次递增。Using ::4 等价 using 1:2:4。如果 只有一个输入列，此值将作为 y，x 由 plot 根据数据条目编号产生（从零开始）。如果数据包含字段名，那么字段名占用第 0 编号，因此数据从 1 开始。如果没有字段名，数据从 0 开始编号。数据字段编号 0 代表条目编号。

数据文件如右图：

using 中可以引用 0 字段，获得自动产生的数字编号。

```
plot 'data.dat' using 1:2:0 with labels
```

执行如上命令，(11,18)点 label 为 0

加上字段名，会发现(11,18)点 label 为 1

using 实例：

普通应用前文已述，如果你有文件各个字段之间不是用空白分割而是使用逗号分割，那么使用 set datafile separator ","。当然还有其他解决办法：

```
plot 'file' using 1:($2+$3) '%lf,%lf,%lf' with points
```

类似 binary format 选项，using 的数据类型表示法和 format 是类似的。

%lf 表示 长浮点类型，%*lf 忽略一个长浮点，%*20[^\n] 忽略 20 个非换行字符。

注意：

```
plot 'file';plot 'file' using 1:2 ;plot 'file' using ($1):($2)
```

在数据中有的行只有 1 列，有的行有多列时，有少许不同：第一个自动为只有 1 列的数产生 x 值，第二个忽略掉只有一列的。第三个对只有一列的行\$2 会得到未定义值，产生错误。

```
Plot 'file' using 1:2
```

会自动忽略所有无效行。但是强烈建议非数据文本以注释形式出现，这将避免错误。

虚拟列(pseudocolumns)：using 的参数可以指定不存在的字段编号，它们有特殊含义，前文所述字段 0 就是一例。

column(0) 等于当前数据集内记录索引，从 0 开始，每当遇到 2 个空白行就重置为 0。

column(-1) 等于当前数据块索引，从 0 开始，遇到 1 个空白行时重置为 0。

Column(-2) 等于数据集索引，与 index 关键字参数一致。

```
set xrange [-1:*]
set yrange [-1:*]
set zrange [0:*]
splot "my.dat2" using -2:-1:0:(stringcolumn(0)) with labels
```

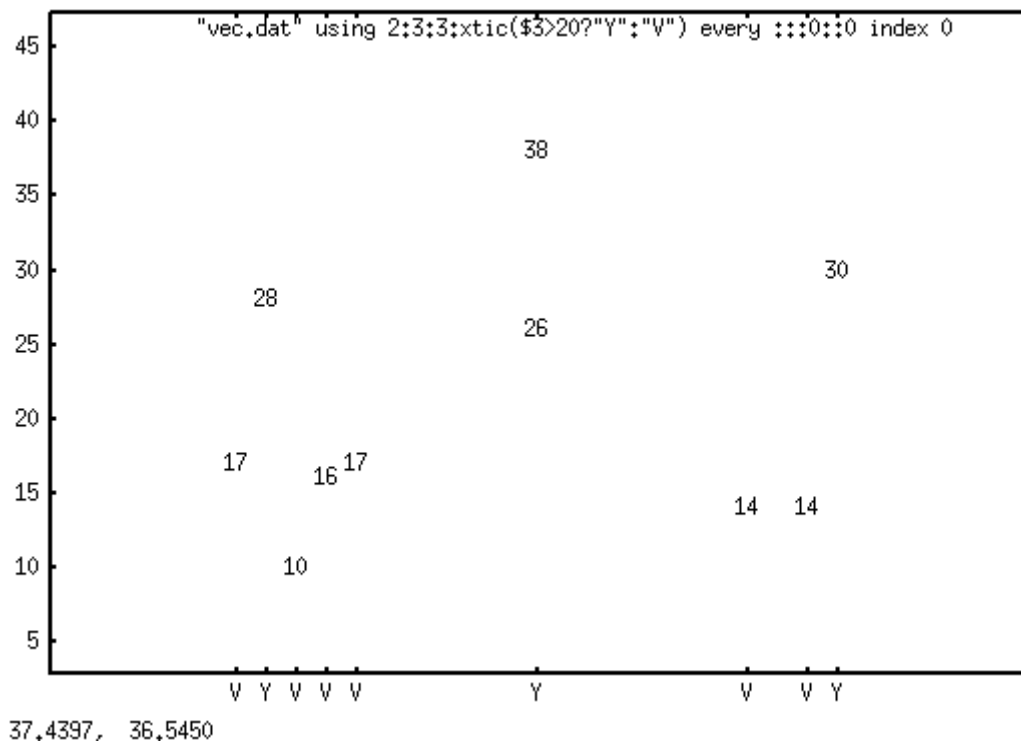
Xticlabels

坐标轴的刻度可以拥有标签。刻度标签使用字符串产生，使用数据字段更常用。如果某个字段值为字符

```
#data.dat
11 18
14 19
18 22
21 25
```

串可以使用 `xticlabels(N)`¹ 或者 `xticlabels(stringcolumn(N))` 来设置对应点的横坐标刻度标签。

`plot "vec.dat" using 2:3:3:xtic($3>20?"Y":""") every :::0::0 index 0 with labels`



为每个点绘制横坐标，值大于 20 的点标签为 Y 否则为 V。与其他的绘制样式结合如 Histograms，参考前文。

`X2ticlabels` `Yticlabels`

`Y2ticlabels` `Zticlabels` 提供其他坐标轴的刻度修改。

Volatile

`volatile` 关键字设置上次读取的数据是从标准输入或者是其他不能重读的文件中。这告诉程序使用 `refresh` 命令代替 `replot` 命令。

函数

`plot` 和 `splot` 可以绘制函数图形。自动对函数采样然后绘制。`Set samples` 和 `set isosamples` 设置了采样精度等设置。

```
f(x,y)=sin($1)*sin ($2)
splot f(x,y) with lines
```

¹ `xticlabels()`有短名：`xtic()`，`yticlabel` 有类似的短名。

parametric

当处于 parametric 模式 , plot 用于绘制函数参数方程。

函数的参数公式:

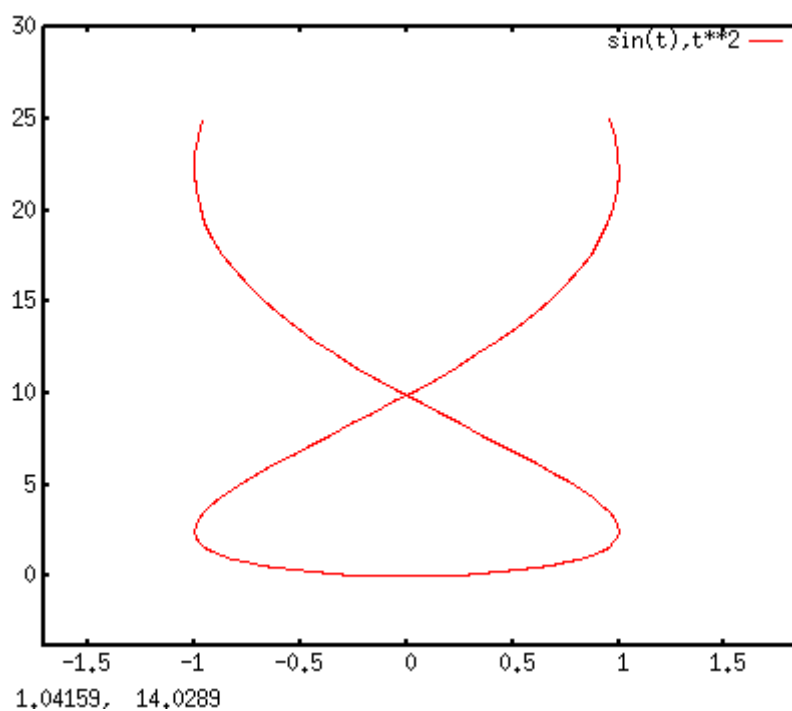
2d $x=f(t), y=g(t)$

3d $x=f(u,v), y=g(u,v), z=h(u,v)$

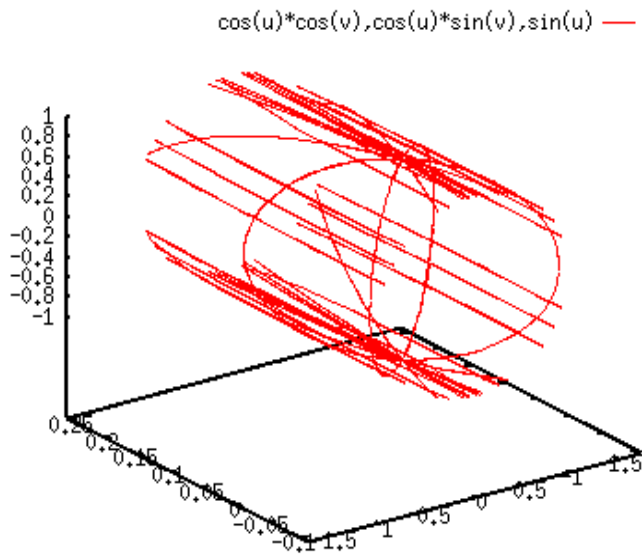
```
set parametric
plot f(t), g(t)
splot f(u, v), g(u, v), h(u, v)
```

实例 :

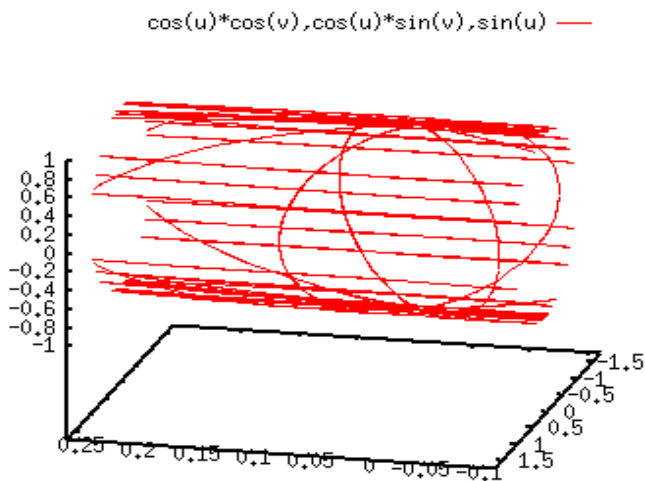
```
set parametric
plot sin(t), t**2
```



```
splot cos(u)*cos(v),cos(u)*sin(v),sin(u)
```



view: 63.0000, 234.000 scale: 1.00000, 1.00000



view: 67.0000, 194.000 scale: 1.00000, 1.00000

范围

范围设置是可选的，用于设置图像显示的横纵坐标范围。与 Set range 有同样效果，只不过如果命令中选择了范围将不使用 set range 值。

语法：

[dummy=min:max]

[min:max]

前文已述。

在非 parametic 模式 , plot [1:2] [3:4] 分别设置 xrange 为[1:2] yrange 为[3:4]

在 parametic 模式 plot [-1:0][1:2] [3:4] 分别设置 trange 为[-1:0] xrange 为[1:2] yrange 为[3:4]

trange 控制了参数的定义域。

x2range y2range 必须使用 set 命令。

Set dummy 控制自变量命名 , 默认自变量为 x , 所以你可以这样写:

```
plot [-pi:pi] sin(x)/cos(x)
```

如果 dummy 设置为 t , 你可以这样写 :

```
plot [t=-pi:pi] sin(t)/cos(t)
```

This sets only the y range, and turns off autoscaling on both axes:

```
plot [ ] [-2:sin(5)*-8] sin(x)**besj0(x)
```

This sets xmax and ymin only:

```
plot [:200] [-pi:] exp(sin(x))
```

This sets the x range for a timeseries:

```
set timefmt "%d/%m/%y %H:%M"
```

```
plot ["1/6/93 12:00":"5/6/93 12:00"] 'timedata.dat'
```

迭代

用于有类似文件名的一系列文件绘制。

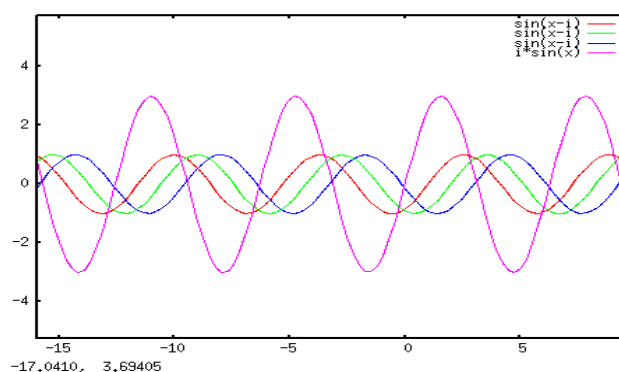
语法 :

```
plot for [<variable> = <start> : <end> {:<increment>}]
```

```
plot for [<variable> in "string of words"]
```

迭代区域以逗号结束 , 逗号之后变量值维持在最后一次迭代。

```
plot for [i=1:3] sin(x-i) , i*sin(x)
```



```
plot for [i=1:3] j=i, sin(j*x) #只绘制了 sin(3*x) 。迭代区没有函数被绘制。  
Plot for [i=1:3] j=i sin(j*x) #绘制了三个曲线 sin(x) sin(2x) sin(3x), 注意逗号的区别
```

```
plot for [dataset in "file1 file2 file3"] dataset . ".dat"
```

绘制三个图形, file1.dat file2.dat file3.dat , 三个文件。 注意 dataset . ".dat" 引号前的小数点为字符串链接运算。

```
list = "apple banana cabbage daikon eggplant"  
item(n) = word(list,n)  
plot for [i=1:words(list)] item[i].".dat" title item(i)  
list = "new stuff"  
replot
```

等价于以下代码：

```
list = "apple banana cabbage daikon eggplant"  
plot for [i in list] i.".dat" title i  
list = "new stuff"  
replot
```

title

关键字 title 设置曲线的标题。Set title 设置整个图片的标题。Plot title 设置的是某个曲线的标题。

语法：

```
title <text> | notitle [<ignored text>]  
title columnheader | title columnheader(N)
```

<text>必须是有效字符串值。有一些选项控制自动产生标题。参阅：datastrings、set key autotitle columnhead。

从输入文件采样曲线标题可以被关键字 notitle 关闭。空标题：title " 等价于 notitle。Notitle 后接字符串参数此字符串被忽略。

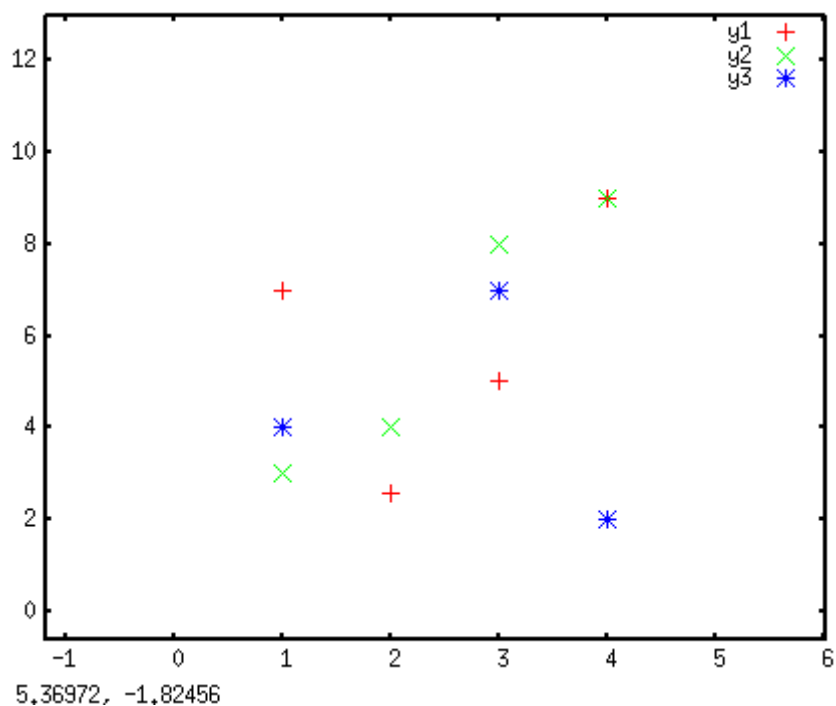
如果 key autotitles 选项打开，并且 title 和 notitle 都没有应用，标题设置为 plot 函数名或者文件名。

title 可以使用 columnhead 关键字，它自动选择迭代变量值引用字段的字段名。如果没有字段名，那么使用数据第一个记录的字段值。

```
plot for [i=2:4] ' vec.tex' using 1:i title columnhead
```

```
#file
x y1 y2 y3
1 7 3 4
2 2.6 4
3 5 8 7
4 9 9 2
```

右图为结果：



with

With 关键字指定绘制样式 和样式选项，使用方法前文已述。现在解释选项：

语法：

```
with <style> { {linestyle | ls <line_style>}
  | {{linetype | lt <line_type>}
  {linewidth | lw <line_width>}
  {linecolor | lc <colorspec>}
  {pointtype | pt <point_type>}
  {pointsize | ps <point_size>}
  {fill | fs <fillstyle>}
  {nohidden3d} {nocontours} {nosurface}
  {palette}}
```

样式：

組 1：

lines points linespoints dots impulses labels steps fsteps
 histeps errorbars errorlines financebars vectors xerrorbar
 xerrorlines xyerrorbars xyerrorlines yerrorbars yerrorlines

組 2：

boxes boxerrorbars boxxyerrorbars boxplot candlesticks

filledcurves histograms image rgbimage rgbalpha circles
ellipses pm3d

第一组样式拥有 线 点 和文本属性，第二组样式还用有填充属性。有些样式拥有子样式。参考绘制样式：29。

默认的样式参数 由 set style function 和 set style data 设置。默认情况，每次 plot 使用一个新的样式参数，参考：**线条样式** 18，和 test 输出

为某条 plot 使用指定样式可以使用语法列出的关键字。比如使用

使用 pointsize 3 使得绘制点的大小为正常的 3 倍，命令行参数会覆盖 set 选项值。Set pointsize 2;plot x w p ps 3; 绘制的点为正常的 3 倍而不是 6 倍。

可以使用 pointsize variable 关键字，使得 pointsize 由输入列设置，因此你需要多一列输入列,参考：**rgb 色彩变量**：21。

如果你的 gnuplot 支持 pm3d 终端，那么关键字 palette 可以用来设置点线面的平滑色彩过度。颜色用来映射值，被称作色彩标尺。

Set palette color 7,5,15

7,5,15 ... traditional pm3d (black-blue-red-yellow)

3,11,6 ... green-red-violet

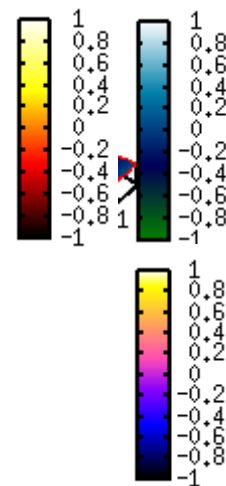
23,28,3 ... ocean (green-blue-white); try also all other permutations

21,22,23 ... hot (black-red-yellow-white)

30,31,32 ... color printable on gray (black-blue-violet-yellow-white)

33,13,10 ... rainbow (blue-green-yellow-red)

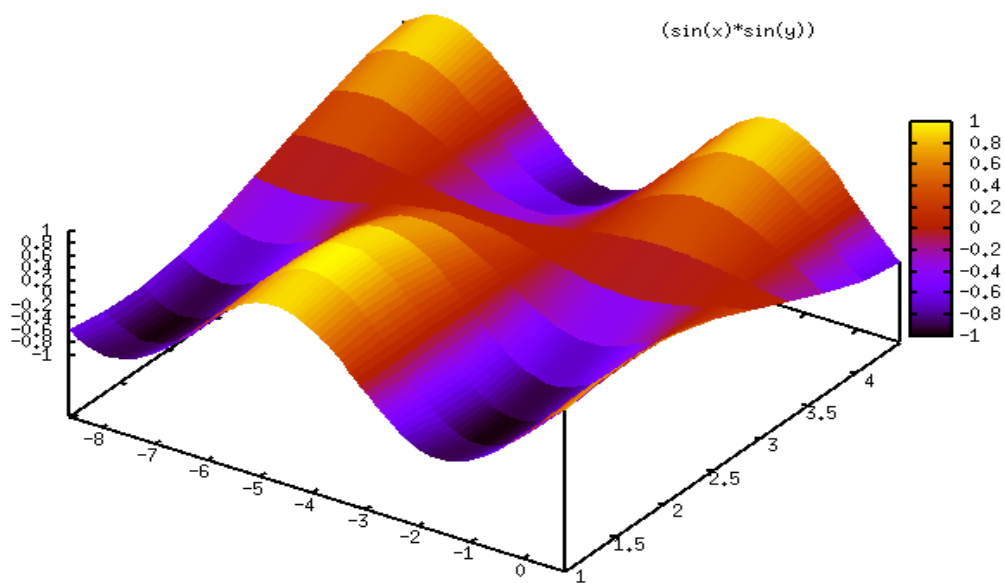
34,35,36 ... AFM hot (black-red-yellow-white)



set palette rgb 23,28,3 #rgbformula 值在[0:36] 每个值定义了一个颜色函数

选择自己合适的标尺使图形更容易阅读。

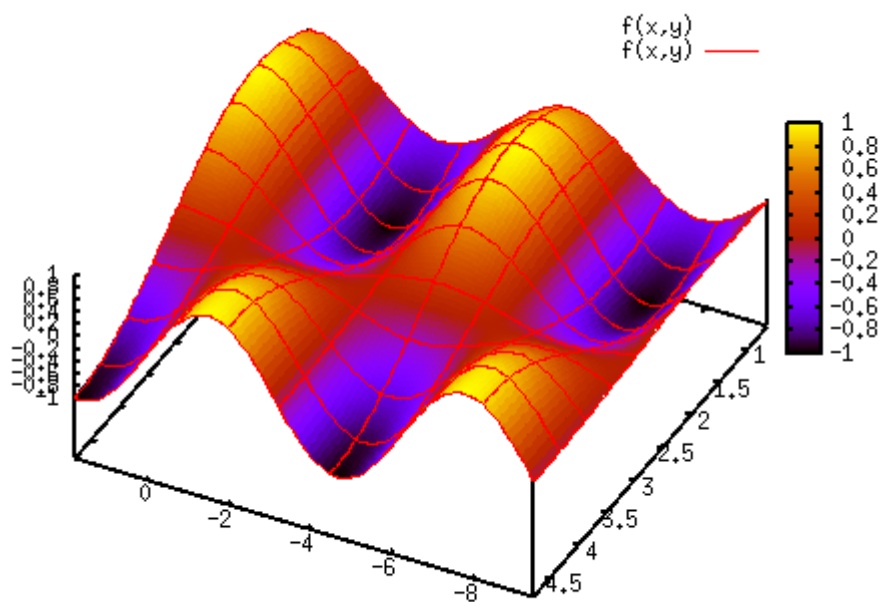
```
splot (sin(x)*sin(y)) with pm3d
```

view: 34.0000, 34.0000 scale: 1.00000, 1.00000

上图图形中存在线条，可以协助眼睛判断空间位置，如果你得到的图像线条影响美观。使用 `set pm3d interpolate 1,1` 修改它的参数，比如 1,0.5 或者 1,5 可以使得图像更平滑。

```
f(x,y)=(sin(x)*sin(y))
set pm3d interpolate 1,10
splot f(x,y) with pm3d
replot f(x,y) with lines
```



view: 31.0000, 207.000 scale: 1.00000, 1.00000

print

print 命令输出参数表达式的值到屏幕。它和 pause 0 <expression> 同意。表达式可以是任何返回数字或者字符串的 gnuplot 表达式。

语法：

```
print <expression> {, <expression>, ...}
```

print 的输出可以被定向到一个文件，set print.

Pwd

pwd 命令输出工作文件夹路径到屏幕上。如果你希望以值的形式保存当前工作目录使用 GPVAL_PWD 变量。参考：show variables all

quit exit

exit 和 quit 命令和 End Of File 字符将退出 gnuplot。在退出之前会清空输出设备(类似 clear 命令)。

Raise

语法：

```
raise {plot_window_nb}
```

raise 命令将指定绘制窗口置顶（相反的操作为 lower）。只有部分输出终端才有绘制窗体：pm win wxt X11。X11 和 wxt 支持多个绘制窗口，默认 raise 提升所有窗口，最近创建的窗口在前面。可选参数在只有一个窗口时被忽略。如果在 x11 终端下没有被提升，可能是绘制窗口运行在不同的 X11 会话下(例如 telnet or SSH)，也有可能是你的窗口管理器搞鬼。

Refresh

reresh 命令类似于 replot，但是有两个不同点。1. refresh 使用已经读取的信息重新绘制。例如在 x11 终端中改变了窗口大小、视图缩放、3d 旋转等使用的就是 refush 指令，它不会反复的从文件中读取信息。2. Refresh 不能添加新的绘制。

实例：

```
plot 'datafile' volatile with lines, '-' with labels
100 200 "Special point"
e
# Various mousing operations go here
set title "Zoomed in view" ; set term post ; set output 'zoom.ps'
refresh
```

如你所见：refresh 重新读取了 '-' 曾经的数据。

Replot

replot 命令没有任何参数时，重新绘制最近一次 plot 或 splot 指令。用于 set 选项发生变化后刷新绘制。当 replot 使用参数时，它的语法与 plot/splot 命令相同，但是不能指定 x y range。它将在 plot 后添加新的绘制参数。

```
Plot sin(x)
replot cos(x) with points
```

等于：

```
plot sin(x), cos(x) with points
```

注意：

```
plot '-' ; ... ; replot
```

这并不好，因为 replot 重新读取 '-'，这使得你需要再次输入一次数据。使用 refresh 更好。replot 指令不能用于 multiplot 模式，因为 replot 只是重复执行了最近的 plot，而不是刷新整个屏幕。

Reread

reread 导致当前 gnuplot 跳转到脚本开始。它可以用于实现循环。将一个 reread 写到一个脚本中，在主脚本中 load 它，你就拥有了一个循环。Reread 应该与 if 命令结合使用。

实例：

Looper file:

```
a=a+1
plot sin(x*a)
pause -1
if(a<5) reread
```

main file:

```
a=0
load 'looper'
```

运行 main 它会绘制 5 次，分别为： $\sin(x)$ $\sin 2x$ $\sin 3x$ $\sin 4x$ $\sin 5x$ ；由于使用的是 plot 指令所以每次只显示一条曲线。

实例：

假设 data 文件拥有 6 个字段。第一个字段是横坐标值，随后 5 个字段是 [0:10] 之间的值。Plotter 脚本包含如下命令：

```

c_p = c_p+1
plot "$0" using 1:c_p with lines linetype c_p
# $0代表 调用参数 0。
if(c_p < n_p) reread

```

main 脚本：

```

n_p=6
c_p=1
unset key
set yrange [0:10]
set multiplot
call 'plotter' 'data'
unset multiplot

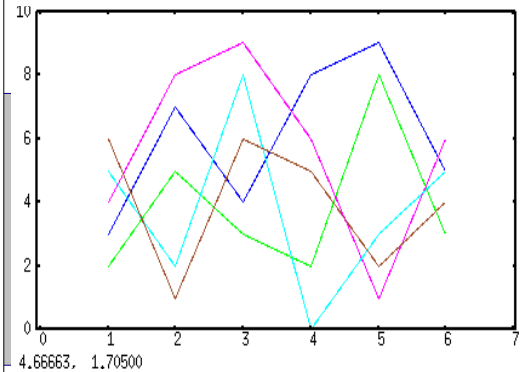
```

#data

```

1 2 3 4 5 6
2 5 7 8 2 1
3 3 4 9 8 6
4 2 8 6 0 5
5 8 9 1 3 2
6 3 5 6 5 4

```



在 gnuplot 中使用 load "main".

当个图像内包含了 5 个曲线。yrange 必须明确设置，这样才能保证 5 条曲线有精确相同的坐标。linetype 必须明确指定，否则所有曲线将使用相同的色彩样式。

Reset

reset 命令重置所有图片相关的 set 选项到默认值。这条命令非常有用。你应该在脚本结束调用此命令，使得你的脚本不会影响到后续脚本。你把设置弄乱了用 reset 恢复。

以下设置不会被 reset：

```

'set term ' 'set output ' 'set loadpath ' 'set fontpath ' 'set linetype '
'set encoding ' 'set decimalsign ' 'set locale ' 'set psdir '

```

reset errors 清除错误状态变量：GPVAL_ERRNO 和 GPVAL_ERRMSG。

Reset bind 恢复所有绑定到默认值。

Save

save 命令保存所有用户定义函数、变量、set term 状态、所有 set 值、所有其他的、最后一次 plot/splot 命令 到指定文件。

语法：

```

save {<option>} ' <filename>'

```

选项可以是：functions、ariables、terminal、set。如果没有指定选项那么使用全部。

save 输出的文件是文本格式，并且可以使用 load 加载。如果没有明确指定 terminal 选项，save 只

会将输出终端保存为注释格式。

特殊文件名 "-", 输出到标准输出。支持管道的系统上可以使用管道符号开始的文件名, 它会被解释成管道操作 save "|grep title >t.gp"

Set/show

set 用于设置选项, show 用于查看选项的值。

Angles

gnuplot 角的单位默认使用弧度, 但也可以使用角度。

```
set angles {degrees | radians}
show angles
```

这会影响 sin cos tan 自变量的单位, 还有坐标系的范围和标签, 也会影响 asin acos asinh acosh 等返回角的函数的返回值。但是不会影响 sinh cosh 的参数, 它们总是使用弧度。

Arrow

使用 arrow 可以绘制任意的箭头。

```
set arrow {<tag>} {from <position>} {to|rto <position>}
    { {arrowstyle | as <arrow_style>}
      | { {nohead | head | backhead | heads}
          {size <length>, <angle> {, <backangle>}}
          {filled | empty | nofilled}
          {front | back}
          { {linestyle | ls <line_style>}
            | {linetype | lt <line_type>}
            {linewidth | lw <line_width> } } } }
unset arrow {<tag>}
show arrow {<tag>}
```

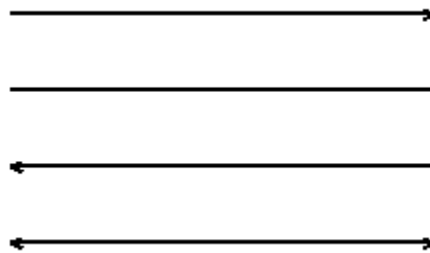
<tag>是一个整数, 它是 arrow 的 ID。如果没有指定 tag, 系统自动设置一个尽可能小的值。ID 用于删除或者设置一个指定的箭头。

<position> 可以使用 x,y or x,y,z 格式。参考: 坐标 6

set from x,y to a,b

表示 箭头由坐标 x,y 指向 a,b。如果使用 rto 代替 to, 那么从 x,y 指向 x+a,y+b。对于线性坐标轴, 坐标起点和终点在图像上的绝对距离取决于它们的相对坐标。对于对数坐标轴取决于变化因子。

Set arrow from 0,0 rto 1,0 #下图使用了 4 个不同选项。



Head 绘制箭头，默认值。




Nohead 关键字使得绘制的箭头没有^尖帽。

Backhead 关键字使得 ^ 出现在箭头开始。

Heads 使得箭头为双向箭头，并不是所有终端都支持双向箭头。

箭头大小由 size <length>,<angle> 或者 size <length>,<angle>,<backangle>控制。

<length>控制箭头分支的长度，单位与 x 轴相同，不过也可以使用 first second graph screen character 关键字。<angle>设置箭头分支的角度（单位是角度不是弧度）。<backangle> 表示次夹角见图，但要注意:<backangle>必然大于<angle>。角 A 等于 $2 * \text{<angle>}$,角 B= $2 * \text{<backangle>}$ 。此选项只在 filled 或 empty 关键字应用时有效。<angle> 大于 90 度就成为了反向箭头。

Filled 填充： 
empty 空心： 
size ,150 filled: 



当 pm3d 样式支持填充。填充和轮廓不被某些终端支持例如：latex metapost tgif。

linewidth , linestyle 参考绘制样式，21。front 关键字使得箭头遮住图表，back 关键字使得图标遮住箭头。

实例：

箭头从 0,0 指向 1,2 line 样式 5：

```
set arrow to 1,2 ls 5
```

设置箭头 ID 为 3，从画布左下角指向-5,5,3::

```
set arrow 3 from graph 0,0 to -5,5,3
```

改变 ID 3 箭头，指向 1,1,1 起点不变，nohead 宽度为 2：

```
set arrow 3 to 1,1,1 nohead lw 2
```

绘制垂直箭头从 横坐标为 3，从图片底到图片顶，应用 nohead。

```
set arrow from 3, graph 0 to 3, graph 1 nohead
```

绘制箭头，并且使得箭头两端成为 T 型。

```
set arrow 3 from 0,-5 to 0,5 heads size screen 0.1,90
```

删除箭头 2：

```
unset arrow 2
```

删除所有箭头：

```
unset arrow
```

所有箭头：

```
show arrow
```

autoscale

自动缩放允许对 x y z 单独开启，或者全部开启。默认缩放所有坐标轴。如果你希望根据图像中的部分元素缩放，你可以把其他对象标记为 noautoscale，参见：plot data 72。

语法：

```
set autoscale {<axes>{|min|max|fixmin|fixmax|fix} | fix | keepfix}
unset autoscale {<axes>}
show autoscale
```

<axes>可以是 x,y,z,cb,x2,y2,xy 关键字 min max（她们在 xy 下不能工作）告诉 gnuplot 缩放只允许改变坐标轴的最小值或最大值。如果没有关键字被指定，缩放所有轴。

fixmin fixmax or fix 关键字被设置。gnuplot 缩放时不会自动对齐到下一个刻度，这样会保留自动搜索到的精确范围，如果设置了 xrange yrange，此选项不会更改已设置的上界或下界。

实例：

只在 y 轴开启缩放：

```
set autoscale y
```

只允许缩放 y 轴，并且只能改变 y 的下界。

```
set autoscale ymin
```

开启 x2 的自动缩放，并且关闭对齐到刻度。使得图像边界维持在函数的精确边界。

```
set autoscale x2fixmin
set autoscale x2fixmax
```

为 x 和 y 开启自动缩放：

```
set autoscale xy
```

为 x, y, z, x2 and y2 轴开启缩放:

```
set autoscale
```

关闭 x, y, z, x2 and y2 轴的缩放:

```
unset autoscale
```

只关闭 y 轴的缩放:

```
unset autoscale z
```

parametric 模式

开启 parametric 模式，在此模式下 x 轴被自动缩放到能够显示整个曲线区域。当然 y 轴在非 parametric 模式也保持这个行为。

数据文件的绘制不受 parametric 选项影响。

当然 set autoscale t 也是支持的，不过这个选项不会有大的效果。

polar 模式

当处在极坐标模式下，xrange 和 yrange 不会直接影响 autoscale。相对的使用 set xrange 限制坐标轴，xrange 和 yrange 会自动调整以匹配它。当然在 set xrange 之后可以设置 xrange 和 yrange 以获得精确的结果。

Bars

set bars 控制在 error bars 结尾和 box 结尾 的短线。

语法：

```
set bars {small | large | fullwidth | <size>} {front | back}
unset bars
show bars
```

small 与 0.0 同义，large 与 1.0 同义。fullwidth 只用于 errorbars¹和 boxplots。它设置短线的宽度与 box 的宽度一致。front 和 back 关键字只用于 errorbars，控制它的填充遮蔽。

Bind

显示当前热键绑定信息。参考：绑定 21。

1 参见：boxplot 31 errorbars 32 52 53 错误：引用源未找到

Bmargin

set bmargin 设置图像在画布底部的边界。

Border

set border 和 unset border 命令控制 plot/splot 图像边界的显示。注意图像边界并不总是和坐标轴位置相同。

语法：

```
set border {<integer>} {front | back} {linewidth | lw <line_width>}
           {{linestyle | ls <line_style>} | {linetype | lt <line_type>}}
unset border
show border
```

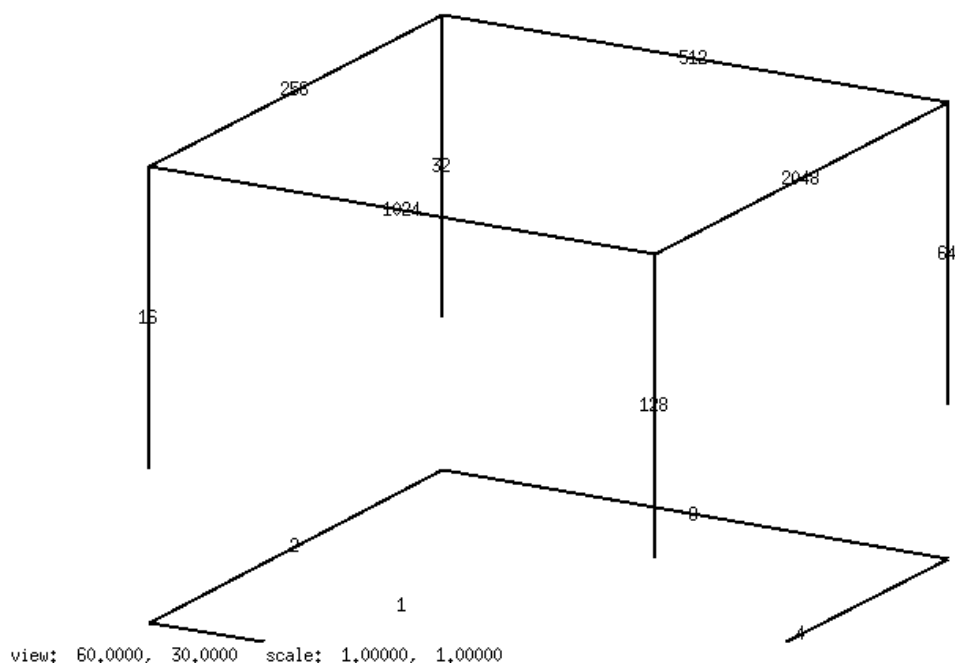
对于 splot 图形可以显示在任意角度。类似 set view 56,103 。x-y 平面的四个方向可以通过 front、back 、 left、 right 关键字引用。

border 使用 12bit 整数编码。最低 4bit 控制 plot 边界和 splot 的基底面。接下来 4bit 控制 splot 竖直方向。最高 4bit 控制 splot 顶部边沿。

图像边界编码		
位	plot	splot
1	bottom 底边界	底面 左前棱
2	left 左边界	底面 左后棱
4	top 上边界	底面 右前棱
8	right 右边界	底面 右后棱
16	无效	左竖直棱
32	无效	后竖直棱
64	无效	右竖直棱
128	无效	前竖直棱
256	无效	顶面 左后棱
512	无效	顶面 右后棱
1024	无效	顶面 左前棱
2048	无效	顶面 右钱棱

最终的编码由她们之和决定。3D 长方体有 12 条边所以需要 12bit 表示。

读者自行体会：



边界是根据视角定义的，无论透视如何旋转，1 总是在底面 左前棱，其他的也一样。

实例：

只显示 plot 左边界和底边界 or splot 底面 左前棱 底面 左后棱。

```
set border 3
```

splot 绘制所有的边界线

```
set border 4095
```

绘制指定的边界线

```
set border 127+256+512 # or set border 1023-128
```

Draw only the top and right borders for a plot and label them as axes:

值绘制顶和右边界坐标轴。x y 不显示

```
unset xtics; unset ytics; set x2tics; set y2tics; set border 12
```

Boxwidth

set boxwidth 控制 box 相关的样式默认宽度：boxes、boxerrorbars、boxplot、candlesticks、histograms。

语法：

```
set boxwidth {<width>} {absolute|relative}
show boxwidth
```

默认情况，每个 box 会扩展到它们互相接触。set boxwidth 用来设置固定宽度。relative 关键字指

定此选项参数作为 默认值的比例因子。

一个 set boxwidth 的参数单位与 x 轴一致。如果 x 轴是对数轴情况有所不同。

设置 box 宽度为自动：

```
set boxwidth
```

使用第 4 输入列：

```
set boxwidth -2
```

同样的功能可以在 using 关键字中使用：

```
plot 'file' using 1:2:3:4:(-2)
```

设置成默认值的一半：

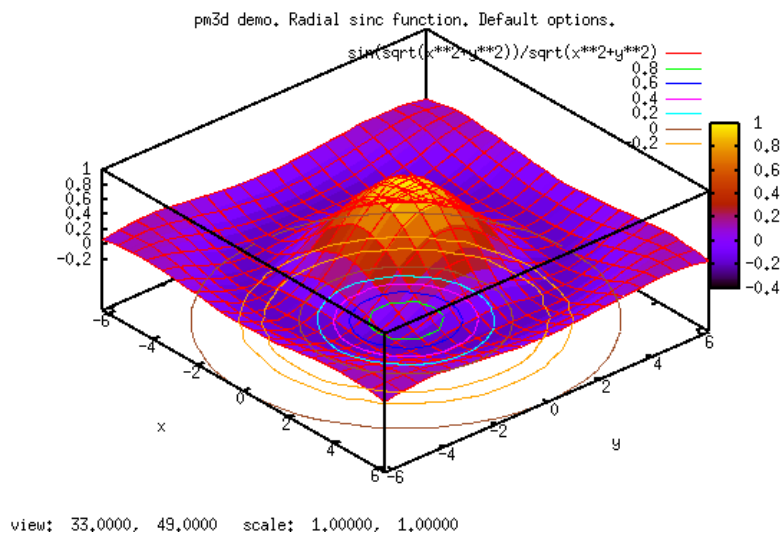
```
set boxwidth 0.5 relative
```

设置成 2 个单位宽。

```
set boxwidth 2 absolute
```

clabel

如果设置了 clabel，gnuplot 将自动为等高线选择 linetype。clabel 选项默认开启。每个 linetype 代表的 z 值会显示在曲线标题。如果关闭此选项，所有等高线使用同一 linetype(曲线 linetype+1)；此选项需要同 set contour 同时使用。



语法：

```
set clabel {'<format>'}  
unset clabel  
show clabel
```

<format> 默认的格式是 "%8.3g", 它显示 3 个小数位。第一个等高线 linetype 是曲线的

linetype+1;

clip

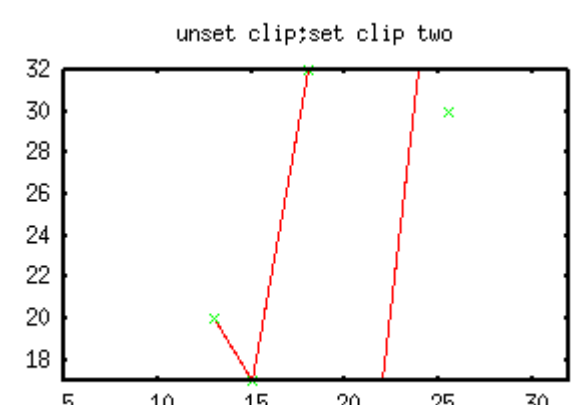
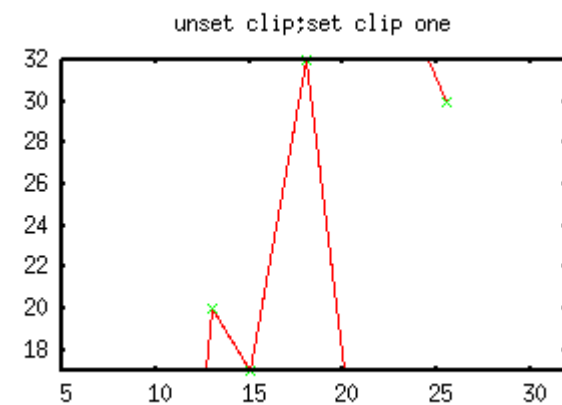
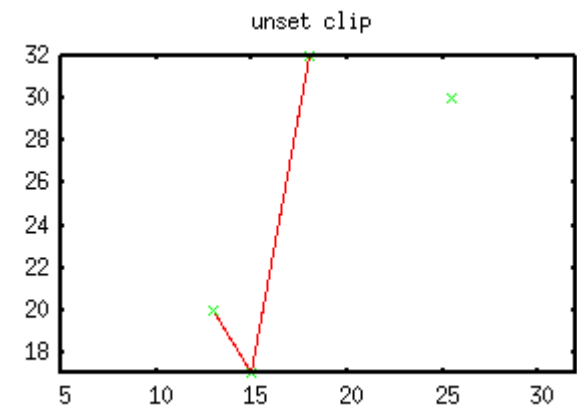
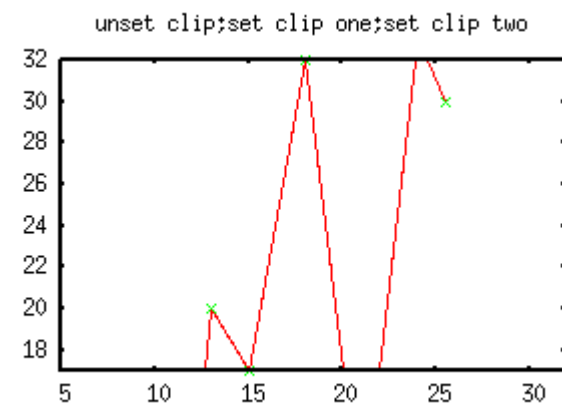
Gnuplot 可以不绘制在端点在图像外的线段和点。

语法：

```
set clip <clip-type>
unset clip <clip-type>
show clip
```

gnuplot 支持三种类型的剪取：points,one,two; 这 3 中类型可以同时在一副图像中开启。注意：pm3d 的剪取不受此设置影响，它受 set pm3d clip1in 和 set pm3d clip4in。

points 剪取告诉 gnuplot 允许绘制在绘图区内但是靠边界非常近的点。可以控制点的符号不会超出图像边界。one 告诉 gnuplot 允许绘制一个端点在图像内，另一个端点在图像外的线段。two 允许绘制两个端点都在图像外的线段。参考下面插图第一副是完整的图像，第二副是关闭所有 clip 的图像。



实例：

关闭所有 clip 选项。

```
unset clip
```

只开启 one clip

```
unset clip;set clip one
```

cntrparam

set cntrparam 控制等高线的生成和平滑。show contour 查看当前 cntrparam 设置。

语法：

```
set cntrparam { {  
    | linear  
    | cubicspline  
    | bspline  
    | points <n>  
    | order <n>  
    | levels { auto {<n>} | <n>  
        | discrete <z1> {,<z2>{,<z3>...}}  
        | incremental <start>, <incr> {,<end>}  
        }  
    } }  
show contour
```

这个选项有两个功能，1. 它决定等高线等高点的决定(等间隔采样或使用采样函数)。2. 它控制等高线绘制方式。<z1><z2> ... <zn> 她们是返回数值的表达式。

liner、cubicspline、bspline 控制间隔和逼近。linear 关键字使用直线连接各个等高点，等高间隔是相等的。cubicspline 分段的线性等高曲线连接。Bspline 使用曲线逼近等高线。

points 这个选项控制 cubicspline 和 bspline 等高线片段使用的点数目，值越大等高线越平滑。

order 关键字控制 bspline 逼近。值越大曲线越平滑（逼近线越平滑那么离原始值就越远）。此选项只用于 bspline。值介于[2:10] 2 等价于 linear。

levels 控制等高线级数，默认由 auto 控制，可以为 auto 指定整数<n>，n 设置一个名义上的值。实际的级数由图形标签的数目决定。如果曲面被 zmin and zmax 限制，等高线将会把 z 值范围分为 n 份。

discrete 指定一个高度列表，gnuplot 只为列表中指定的高度绘制等高线。

incremental 根据 开始高度，高度间隔，等高线数目 产生的列表绘制等高线。<end>被用来决定等高线级数。但是它会被 set cntrparam levels <n>修改。

实例：

```
set cntrparam bspline
set cntrparam points 7
set cntrparam order 10
```

设置等高线为自动 5 等级。

```
set cntrparam levels auto 5
```

为三个高度绘制等高线.1, .37, 和 .9:

```
set cntrparam levels discrete .1,1/exp(1),.9
```

指定一个高度列表：0~4 间隔为 1

```
set cntrparam levels incremental 0,1,4
```

设置等高级数为 10(会改变 incremental <end> or 自动级数):

```
set cntrparam levels 10
```

设置等高线从 100 开始间隔 50，级数受到 level 的控制。

```
set cntrparam levels incremental 100,50
```

参考：contours.dem 和 discrete.dem 。 set contour , set clabel。

color box

pm3d palette 色彩标尺受到 color box 的控制。

```
set colorbox
set colorbox {
    { vertical | horizontal }
    { default | user }
    { origin x, y }
    { size x, y }
    { front | back }
    { noborder | bdefault | border [line style] }
}
show colorbox
unset colorbox
```

color box 的初始位置可以是用户定义的或者默认值。box 可以使用 orgin 选项和 size 选项。box 可以绘制在图型前方(front)，也可以绘制在图形后方(back)。标尺色彩渐变方向 vertical 竖向（默认），horizontal 横向。

origin 和 size 关键字，与 user 关键字同时使用有效。x y 的单位默认为 screen¹。

```
set colorbox horiz user origin .1,.02 size .8,.04
```

设置色彩标尺在底部，并且水平方向渐变。

border 使得标尺拥有边框（默认）。noborder 关键字关闭边框。Border 关键字后方接正整数表示使

¹ screen 参见 坐标 6，坐标注解：7

用的线条样式。

```
set style line 2604 linetype -1 linewidth .4
set colorbox border 2604
```

设置 ID 为 2604 的线条样式 线条类型¹为 -1 宽度为 4。

Contour

set contour 开启等高线。它只对 splot 有效，依赖 grid data。如果希望等高线从无网格数据产生请使用 : set dgrid3d 可以创建一个合适的网格。

语法：

```
set contour {base | surface | both}
unset contour
show contour
```

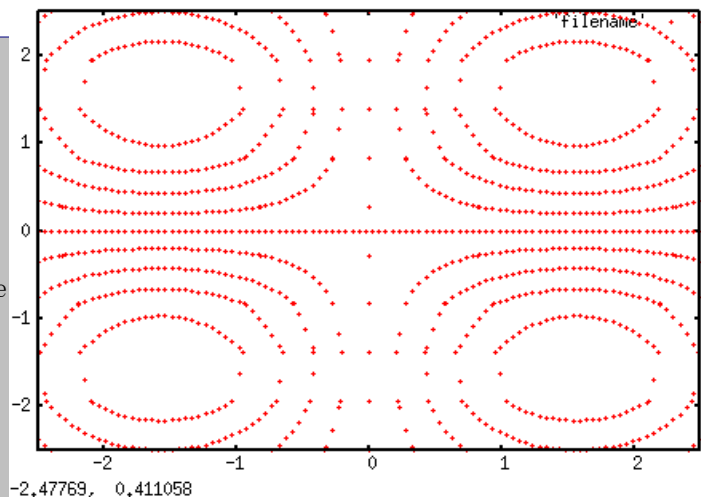
base 在 xy 平面上绘制等高线的投影。surface 在曲面上绘制等高线。both 同时在 base 和 surface 绘制等高线。

参考 cntrparam 101 控制等高线。

可以 unset surface 选项，使得图像只出现等高线。也可以控制等高线信息写入一个文件中，这样可以把等高信息作为 2d 数据读取。

```
set xrange [-2.5:2.5]
set yrange [-2.5:2.5]
unset surface
set contour
set cntrparam level 10
set table 'filename'
splot sin(x)*sin(y) with pm3d notitle
unset table
# 现在等高数据保存在 filename

set term x11 persist
plot 'filename' #你可以使用 smooth
bezier 将点连起来。
reset
```



为了绘制等高线，等高数据会作为网格数据组织。所有拥有相等高度的点被输出在一起，随后是另单独的空白行用于分割拥有相等高度的多根曲线，连续的 2 个空白行用于分割下一个高度的等高点。执行上面的代码，看一下 filename 你就明白了。

1 line style 参见 test 命令：18

Datafile

Set datafile 命令控制 plot/splot/fit 命令对字段的解析。

set datafile fortran

允许字段内出现 fortran D Q 常量。这个功能会降低文件处理速度。并且请在你确认你的数据包含 fortran D Q 常量时使用。使用 unset datafile fortran 关闭此选项。

set datafile nofpes_trap

告诉 gnuplot 不要重新初始化浮点异常，除了在数据文件表达式计算之前。这对快速处理非常大的文件有意义。不过 gnuplot 会在浮点异常发生时终止。

set datafile missing

此命令告诉 gnuplot 用什么字符串表示缺失数据。这个字符串具体有何意义取决于 using 关键字。

语法：

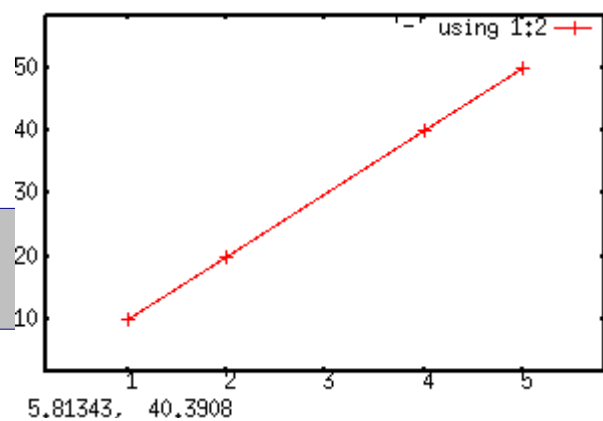
```
set datafile missing {"<string>"}
show datafile missing
unset datafile
```

实例：

```
# 忽略 IEEE NaN ("Not a Number") 值
set datafile missing "NaN"
```

实例：

```
set style data linespoints
plot 'data'
set datafile missing "?"
plot 'data'
plot 'data' using 1:2
plot 'data' using 1:($2)
```

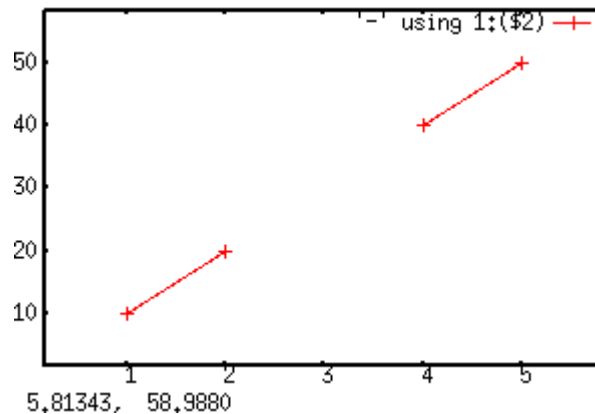


#Data
1 10
2 20
3 ?
4 40
5 50

第一条 plot 会把 3 ? 行处理为：数据只包含一个有效值，使用 \$0 作为 x \$1 作为 y。第二个 plot 和第三个 plot 含义一致，他会忽略无效行 "3 ?"。并且认为它相邻的点连续。

第四个 plot 会忽略无效行 "3 ?", 认为它相邻的点不连续。

missing 选项没有默认值。所有无法被解析的数据都当作是 missing 数据。



set datafile separator

设置数据文件字段分隔符。默认使用 空白字符(空格 和 tab)。当读取 csv (逗号分割值) 文件时你需要此选项。

语法:

```
set datafile separator {"<char>" | whitespace}
```

实例:

输入文件以横向制表符分割字段。

```
set datafile separator "\t"
```

输入文件以逗号分割字段

```
set datafile separator ",",
```

set datafile commentschars

设置数据文件注释符号。默认数据文件注释符号在 vms 上为 "#!" 其他系统中为 "#". Gnulot 自动忽略以注释符号为第一个非空白字符的行。

语法 :

```
set datafile commentschars {"<string>"}
show datafile commentschars
unset commentschars
```

默认值 "#!" on VMS and "#" otherwise.

所以下面的行会被完全忽略 :

1 2 3 4

但是下面的行 :

1 # 3 4

会产生错误数据

```
set datafile missing ' #'
```

设置此选项就可以了

实例：

```
set datafile commentschars "#!%"
```

set datafile binary

设置二进制文件默认参数。语法与 plot splot binary 关键字相同。参见：plot binary P.67

decimalsign

set decimalsign 命令选择在 label 或者坐标轴 label 输出数字时使用的十进制标记（小数点符号）。

语法：

```
set decimalsign {<value> | locale {"<locale>"}}  
unset decimalsign  
show decimalsign
```

参数<value>是字符串，它显示在小数点的位置。常见的选择是 "." 句点 或者 "," 逗号。但是其他的选择也是很有用的。如果你省略<value>参数，选项恢复到默认值。

实例：

设置逗号为小数分隔符，符合欧洲国家习惯。

```
set decimalsign ' , '
```

如果你设置了精确字符串，那么它只会影响 gprintf() 函数返回值（包括坐标轴标签），不会影响 sprintf 函数。并且它不改变输入文件的小数格式。如果想同时改变输入和输出格式使用：

```
set decimalsign locale
```

此命令告诉 gnuplot 根据环境变量 LC_ALL, LC_NUMERIC, LANG 设置输入和输出小数格式。

```
set decimalsign locale "foo"
```

此命令告诉 gnuplot 根据本地语言配置文件 "foo" 设置输入输出小数格式（你得确认它存在）。如果 foo 不存在会显示错误信息，并且使用默认小数格式。linux 系统允许通过 "locale -a" 得到当前支持的本地化配置文件列表。它应该是这种格式：zh_CN.utf8（简体中文 utf8 编码）。当然本地化也需要 C 运行库的支持。因系统而异。

```
Set decimalsign locale ; set decimalsign "."
```

此命令将输入输出格式设置为本地格式，随后设置输出格式为 句点。

Dgrid3d

Set dgrid3d 命令开启和设置非网格数据到网格数据的映射。参考：splot grid_data, 得到网格数据

的详细信息。

语法：

```
set dgrid3d {<rows>} {,<cols>}}
    { splines |
      qnorm {<norm>} |
      (gauss | cauchy | exp | box | hann)
      {kdensity} {<dx>} {,<dy>} }
unset dgrid3d
show dgrid3d
```

dgrid3d 选项默认关闭。如果打开，读取的 3d 数据总是被作为分散数据。网格大小取决于分散数据的边界和 <rows> <cols> 参数。网格在指定的行(x) 和列(y)上是等距离的。z 值根据权重决定和分散数据 z 值样条插值。换句话说，规则的网格被创建，并且为数据的行产生平滑逼近。逼近曲线绘制在数据行所在的位置。

如果忽略<cols>则它等于<rows>，<rows>默认为 10。

Gnuplot 为数据逼近提供了多个算法。有些算法需要附加的输入列。

splines 算法计算使用薄板样条计算插值。它不需要附加输入列。

qnorm 算法计算输入数据在每个网格点的平均权重。每个数据点的权重计算公式为：

dx dy 是网格点到每个数据点的距离，<norm>对于某些 qnorm 是 2、4、8 或 16。使用欧几里德距离法在权重方面最优化($(dx^2 + dy^2)^{norm}/2$)，当然任何非负数都可以被设置。<norm>的幂可以被设置为一个可选参数。

最后，需要指定一些平滑算法去计算平均权重： $z = \text{Sum}_i w(d_i) * z_i / \text{Sum}_i w(d_i)$, z_i 是第 i 数据点的 z 值， d_i 是第 i 数据点到当前网格点的距离。距离越近数据点的权重越高，不同算法有不同计算方法。

可用的算法：

gauss :	$w(d) = \exp(-d*d)$	
cauchy :	$w(d) = 1/(1 + d*d)$	
exp :	$w(d) = \exp(-d)$	
box :	$w(d) = 1$	if $d < 1$
	$= 0$	otherwise
hann :	$w(d) = 0.5*(1-\cos(2*\pi*d))$	if $d < 1$
	$w(d) = 0$	otherwise

使用这些算法时，最多可以使用 2 个附加的 数字参数，dx dy。它们用于修正距离公式：

$d_i = \sqrt{((x-x_i)/dx)^2 + ((y-y_i)/dy)^2}$)。x y 是当前网格点坐标， x_i y_i 是当前

数据点坐标，默认 $dx == dy == 1$ 。这个功能用于控制距离单位为数据本身的单位。

可选的关键字 `kdensity2d`，必须在权重算法后面出现，必须在 `dx dy` 因子前方，它导致网格点不会被 $\sum_i w(d_i) * z_i$ 除。如果所有 z_i 都相等，这是一个有效的双变核密度估计。：核函数(前面的 5 个算法)被应用于每个数据点。随后为每个网格点计算核值之和，然后平滑曲面被绘制（只绘制曲面，不再绘制数据点）。这类似与 `smooth kdensity` 选项被设置到 1 维数据。（参阅：`kdensity2d.dem` 实例）

查看：`drid3d.dem scatter.dem`

dummy

`set dummy` 设置默认自变量名。

语法：

```
set dummy {<dummy-var>} {,<dummy-var>}  
show dummy
```

在 `parametric` 参数公式模式和 `polar` 极坐标模式默认自变量名为 `t`，其他情况默认为 `x`。类似的 `splot` 在这两个模式下自变量为 `u v`，其他情况为 `x y`。

这个功能对绘制物理函数提供方便，它们通常不是使用 `x y` 作为自变量。

例如，跟时间相关的函数：

```
set dummy t  
plot sin(t), cos(t)
```

这个命令至少需要一个参数，否则出现错误提示。

实例：

```
set dummy u,v #自变量设为 u,v  
set dummy ,s #修改第二个自变量为 s。
```

Encoding

`set encoding` 命令选择字符编码。

语法：

```
set encoding {<value>}  
set encoding locale  
show encoding
```

可用的选项为：

`default` 默认编码

`iso_8859_1` - 西欧常用的编码。在 PostScript 上称为 ISO-Latin1。

`Iso_8859_15` - 是 `iso_8859_1` 的变体包含欧洲字符

iso_8859_2	- 用于中欧和东欧
iso_8859_9	- 用于 Turkey , 也被称作 Latin5
koi8r	- 用于 Unix 西里尔编码
koi8u	- 用于 乌克兰 Unix 西里尔编码
cp437	- MS-Dos 代码页
cp850	- OS/2 代码页, 西欧。
cp852	- OS/2 代码也, 中欧和东欧。
cp950	- MS Big5 编码 (只用于 emf 终端)
cp1250 , cp1251 , cp1254	
sjis	- 日文编码
utf8	- 可变长度多字节编码, 支持多国语言, linux 的默认编码。

Set encoding locale 与其他选项不同。它根据环境变量 LC_ALL、LC_CTYPE、LANG 决定编码。这个功能很重要比如你向 wxt 终端输出以 utf8 或者其他编码的字符, gnuplot 必须以同样的编

$$1/(dx^{norm}+dy^{norm})$$

码处理它。这个选项不影响数据和日期的本地化设置, 也不影响数字的本地设置。

参见: set decimalsign 106 , set locale。

通常应该在指定输出类型之前设置此选项。注意: 有些终端类型不支持某些编码。

Fit

Set fit 控制 fit 命令的日志输出。

语法:

```
set fit {logfile {"<filename>"}} {{no}errorvariables} {{no}quiet}
unset fit
show fit
```

<filename> 是字符串参数。如果没有提供<filename>,或者使用了 unset fit,那么输出文件为 fit.log 或者是 环境变量 FIT_LOG 的值。如果<filename>以 / 或者 \ 结束, 那么被当作文件夹路径,日志文件 fit.log 将位于文件夹下。

如果 errorvariables 选项开启。对于每个待拟合参数误差值被保存到, 拟合变量名+_err 中 (a_err b_err)。

这主要用于在图形上标注误差值。

```

set fit errorvariables
fit f(x) 'datafile' using 1:2 via a, b
print "error of a is:", a_err
set label 'a=%6.2f', a, '+/- %6.2f', a_err
plot 'datafile' using 1:2, f(x)

```

默认时误差信息写在日志里，同时显示在终端会话中。Set fit quiet 关闭输出。

Fontpath

fontpath 设置定义了 gnuplot 搜索字体文件的路径。目前只有 Postscript 终端支持 fontpath。如果字体文件不能在当前目录下找到，那么搜索 fontpath 路径。

语法：

```

set fontpath {"pathlist1" {"pathlist2"...}}
show fontpath

```

目录名可以是单个路径，也可以是用系统路径分隔符分割的路径列表，unix 路径分隔符为冒号，dos/windows/OS2 平台使用分号。show pathpath 和 save 命令输出的目录会自动替换这些分隔符为空格。如果一个路径以感叹号!结尾，那么会对这个路径进行递归搜索。

环境变量 GNUPLOT_FONTPATH 的值被添加到 fontpath 后面。如果此环境变量没有定义那么使用系统默认文件夹。当然第一次调用 set fontpath、show fontpath、save fontpath、plot、splot 嵌入字体时会花点时间。

show fontpath 显示当前的字体搜索路径，包括系统默认路径。但是 save 命令只保存用户定义的字体路径。

通过 gdlb 库访问字体的输出终端，依靠 GDFONTPATH 环境变量搜索字体。

Format

format 控制坐标轴刻度标签，可以使用 set format 或者 set tics format，或者 set {axis}tics format。

语法：

```

set format {<axes>} {"<format-string>"}
set format {<axes>} {'<format-string>'}
show format

```

<axes>可以是 x y xy x2 y2 z cb 或者 省略（应用在所有坐标轴）。一下的设置等价：

```

set format y "%.2f"
set ytics format "%.2f"

```

<format-string>长度限制在 100 个字符以内。默认的输出格式为"%g"。在 latex 终端下使用\$%g\$，你可以使用 tex 的数学环境。如果未指定<format-string>，恢复成默认值，如果设置为空字符串""，那么没有刻度标签刻，但是刻度依然会显示。使用 unset xtics 或 set tics scale 0，会导致刻

度不显示。

格式化字符串中可以使用转义字符，也可以使用增强字符模式，此时需要使用双引号。格式化字符串参见 `gprintf` .

参考：electron.dem

gprintf

类似 C 语言的 `sprintf` 函数，`gprintf` 为 `gnuplot` 使用的格式化函数，它提供了 `gnuplot` 内置的一些功能，但是格式化占位符上有所区别。`Gnuplot` 自动输出的标签也调用了 `gprintf` 函数。

格式化占位符

如果不处在日期模式下支持以下符号：

刻度标签数字格式化占位符	
占位符	含义
%f	浮点数，类似与 C 语言
%e or %E	科学记数法：2.3*e3
%g or %G	根据数字大小自动选择 %g 和%f
%x or %X	hex 16 进制
%o or %O	octal 8 进制
%t	基于 10 的尾数
%l	基于对数标尺的底数的尾数
%s	基于对数标尺的底数的尾数； 科学指数
%T	以 10 为底的指数
%L	以对数标尺底数为底的指数
%S	科学记指数
%c	用字母表示当前科学记数法的指数： k
%b	ISO/IEC 80000 国际标准单位尾数(ki, Mi, Gi, Ti, Pi, Ei, Zi, Yi)
%B	ISO/IEC 80000 国际标准单位前缀 (ki, Mi, Gi, Ti, Pi, Ei, Zi, Yi)
%P	pi 的倍数

一个科学指数是指数是 3 的倍数的指数。科学指数的字符表示%c 在-18 到 18 之间有效。超出这个范围将从头开始循环。

如果你对本段不理解请上网查询,"c printf 函数", 参考它的解释：

在%号后面和字母前面可以出现一些修饰符例如：% -8.3f 。“-”表示左对齐，“+”表示为数字添加正符号。' ' 空格，对于正数 在符号位上使用空格占位。# 表示，在所有小数的小数部分都使用小数点，即使小数部位全为 0。一个正数表示 整体输出占位的最低宽度。数字 0 表示字段宽度不足使用 0 填充。小数点后接非负数，表示小数部分最低精度。

实例：%-10.3f 表示 左对齐 ，最低 10 个字符宽度，小数部分保留 3 位。

命令	产生字符串
print gprintf("\%-10.3f",3.1234)	=> '3.123 '
print gprintf("\%#-10.3f",3)	=> '3.000 '
print gprintf("\%#10.3f",3)	=> ' 3.000'
print gprintf('%+010.3f',3.1234)	=> '+00003.123'

实例:

```
set format y "%t"; set ytics (5,10)           # "5.0" and "1.0"
set format y "%s"; set ytics (500,1000)       # "500" and "1.0"
set format y "%+-12.3f"; set ytics(12345)     # "+12345.000 "
set format y "%.2t*10^+03T"; set ytic(12345)  # "1.23*10^+04"
set format y "%s*10^{%S}"; set ytic(12345)    # "12.345*10^{3}"
set format y "%s %cg"; set ytic(12345)       # "12.345 kg"
set format y "%.0P pi"; set ytic(6.283185)   # "2 pi"
set format y "%.0f%%"; set ytic(50)          # "50%"
set log y 2; set format y ' %l' ; set ytics (1,2,3) #displays "1.0", "1.0" and
"1.5" (since 3 is 1.5 * 2^1)
```

时间日期格式化占位符

在日期模式下允许使用如下占位符：

时间日期格式化占位符	
占位符	含义
%a	星期的单词缩写
%A	星期的单词全写

%b or %h	月份单词缩写
%B	月份单词全写
%d	月份数字 01-31
%D	与格式 %m/%d/%y 等价，只用于输出
%F	与格式 %Y-%m-%d 等价，只用于输出
%k	小时 0-23，一个或者两个数字
%H	小时 00-23，始终两个数字
%l	小时 1-12
%I	小时 01-12
%j	一年当中某天，1-366
%m	月份 01-12
%M	分钟 0-60
%p	ap or pm
%r	"%I:%M:%S %p" 的简写，只用于输出
%R	%H:%M 的简写，只用于输出
%S	秒数，0-60，输出是整数，读入是小数
%s	从 2000 年到现在的秒数
%T	%H:%M:%S 的简写，只用于输出
%U	一年的第几星期。从星期日开始
%w	星期的第几天，0-6 (0 表示星期天)
%W	一年的第几星期，从星期一开始。
%y	公元年低 2 位，0-99，有效年份：1969-2068
%Y	完整 4 位数字公元年。

%S 格式可接受精度修饰，这样可以输出小于 1 秒的值。输出的总字符数目有 24 个字符限制，过长会自动截断。

实例：

假设数据为："76/12/25 23:11:11". Then

```
set format x          # 默认产生 "12/25/76" \n "23:11"
set format x "%A, %d %b %Y" # "Saturday, 25 Dec 1976"
```

```
set format x "%r %D" # "11:11:11 pm 12/25/76"
```

假设数据为："98/07/06 05:04:03.123456". Then

```
set format x "%ly/%2m/%3d %01H:%02M:%06.3S" # "98/ 7/ 6 5:04:03.123"
```

functions

show functions 显示所有用户定义的函数。

语法：

```
show functions
```

关于函数语法更多的信息参见：表达式 9。

参见实例：spline.dem airfoil.dem

grid

set grid 命令允许在绘制图形上绘制网格。

语法：

```
set grid {{no} {m}xtics} {{no} {m}ytics} {{no} {m}ztics}
        {{no} {m}x2tics} {{no} {m}y2tics}
        {{no} {m}cbtics}
        {polar {<angle>}}
        {layerdefault | front | back}
        { {linestyle <major_linestyle>}
          | {linetype | lt <major_linetype>}
          {linewidth | lw <major_linewidth>}
        { , {linestyle | ls <minor_linestyle>}
          | {linetype | lt <minor_linetype>}
          {linewidth | lw <minor_linewidth>} } }

unset grid
show grid
```

网格可以为任何一个坐标轴的主刻度和次刻度开启。可以分别为主刻度和次刻度网格线设定 linetype 和线条宽度。有些输出终端有默认线条样式。

极坐标网格可以在 2d 绘制中开启：一个个同心圆按照刻度间隔绘制，并且角度线也可以根据指定的间隔值绘制（间隔值的单位可以为角度或者弧度，这依赖于 set angles ）。在极坐标模式下，网格不再自动产生。

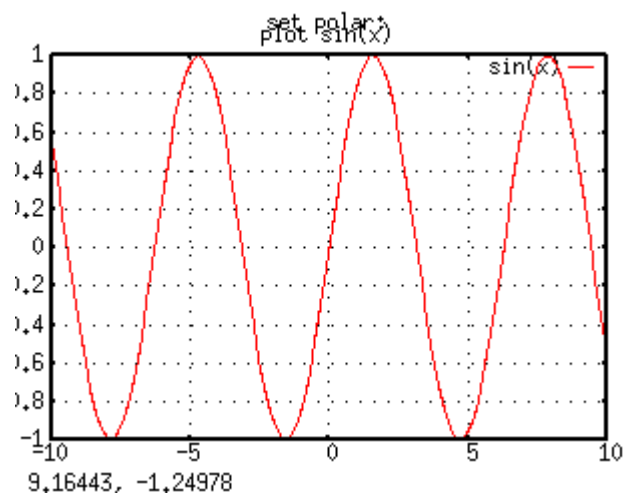
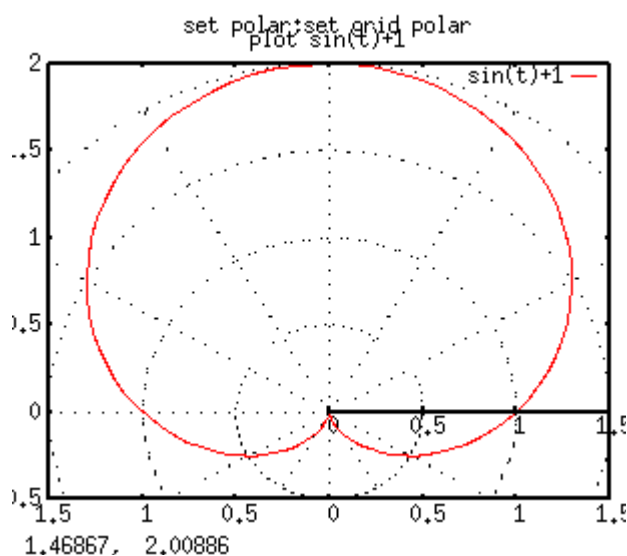
你最好在 set grid 之前设置合适的刻度，gnuplot 不会为不存在的刻度绘制网格线，但是如果你在设置网格之后新增了刻度，那么它的网格线依然会被绘制。

次刻度网格线设置线条样式默认和主刻度网格线样式保持一致。默认的极坐标角度线间隔 30 度。

如果使用 front 关键字那么网格绘制在图像的上方，back 相反。Front 使得网格线不会因为数据太密

集而混在一起。

2d 默认值为 back，3d 默认把图像和网格放在两个图层，一个在前一个在后。由于 hidden3d 选项拥有自己的图层排序方法，所以 set grid front 和 back 在 hidden3d 下无效。



Hidden3d

`set hidden3d` 命令使得 `splot` 图形之间存在 3d 遮蔽，坐标轴也会被图形遮蔽。被遮蔽部分实际上被遮蔽算法移除了。一些可选的特性可以被 `hidden3d` 控制。

语法：

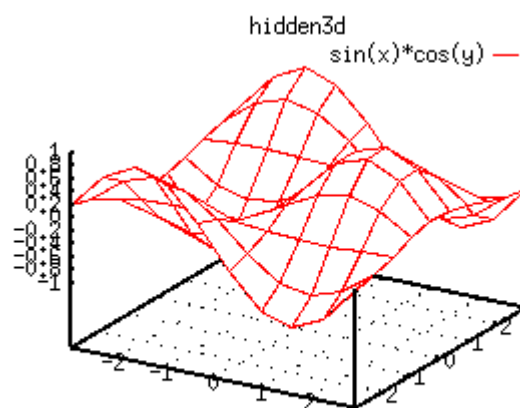
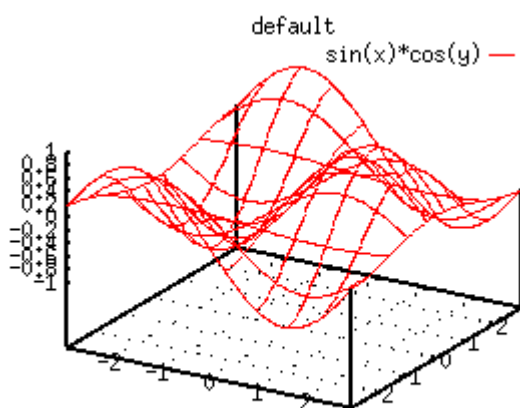
```
set hidden3d {defaults} |
    { {front|back}
      {{offset <offset>} | {nooffset}}
      {trianglepattern <bitpattern>}
      {{undefined <level>} | {noundefined}}
      {{no}altdiagonal}
      {{no}bentover} }
unset hidden3d
show hidden3d
```

隐藏选项对待函数曲面和数据网格为真实平面，她们之间会互相遮蔽。为了让 `hidden` 能工作，曲面必须拥有网格结构（不是大量零散的点组成的点集合），并且必须使用 `with lines` 或者 `with linespoints` 绘制。

如果 `hidden3d` 开启，那么等高线¹、函数曲面、图像边界线²、坐标轴都可以被遮蔽。此时等高线不会出现曲面上，`set contour surface` 不会有作用。

¹ 等高线参见：`set contour 103`。

² 边界线参见：`set border 97`



view: 59.0000, 31.0000 scale: 1.00000, 1.00000 view: 59.0000, 31.0000 scale: 1.00000, 1.00000

标签 (`set label`) 和箭头 (`set arrow`) 永远不会被遮蔽。gnuplot 4.6 `hidden3d` 对于绘制样式 `points`、`labels`、`vectors`、`impulses` 甚至在图像上没有图形的元素都会有遮蔽效果。在绘制选项 `with` 后使用 `nohidden3d` 关键字可以为某个图形关闭遮蔽选项。

`hidden3d` 选项不会导致 `pm3d` 平面产生遮蔽。使用 `set pm3d depthorder` 产生类似效果。混合使用 `hidden3d` 选项和 `pm3d` 样式，1. `set hidden3d front` 2. 使用 `with lines lt -2` 绘制曲面但是不显示曲面，这个隐形曲面在 `hidden3d` 计算中发挥作用。3. 最后你需要再次以 `pm3d` 绘制一次曲面。

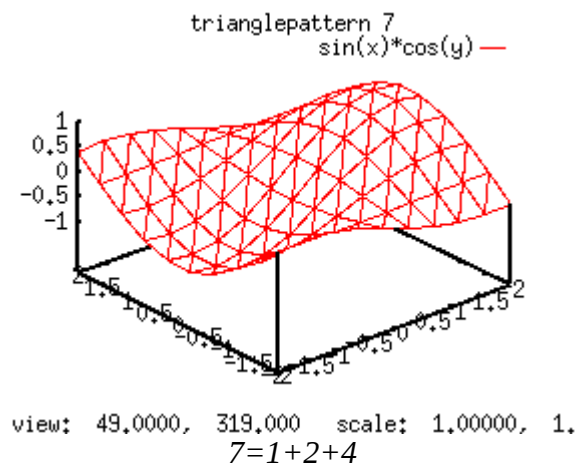
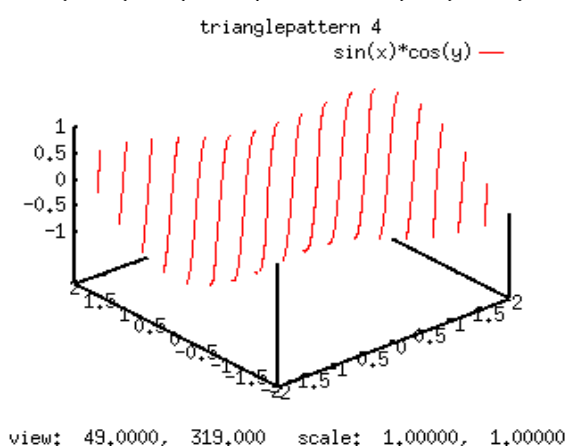
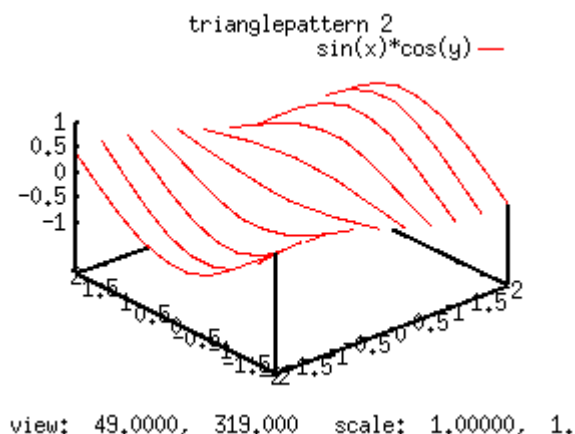
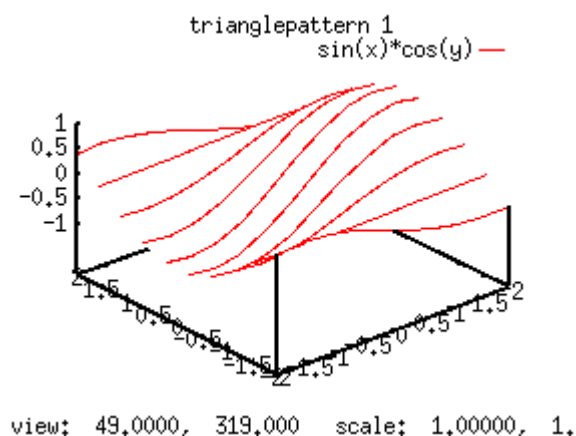
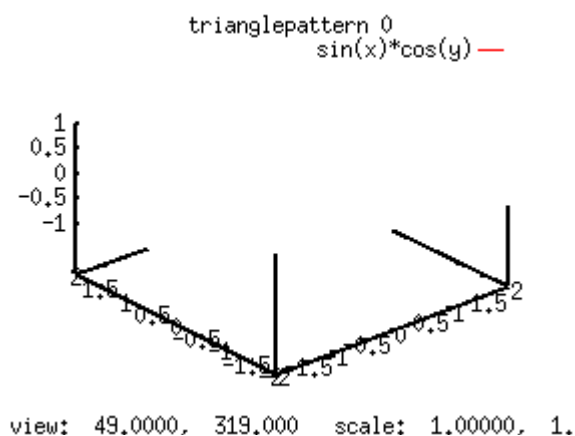
函数值在每个采样点被计算。线性插值算法决定曲线可见部分各个采样点之间的距离。这意味着函数在 `hidden3d` 和 `nohidden3d` 下看起来可能略有不同，因为两个曲线有不同的可见部分。参考：`set isosamples` 和 `set samples`。

用来删除被遮蔽部分的算法有一些特性，可以被 `set hidden3d` 控制。使用 `defaults` 关键字设置选项到默认值。如果没有使用 `defaults` 选项，只有明确出现的选项受到影响。

`offset` 关键字影响在曲面后方(`back`)的线条样式。通常，后方的线条样式值比它前方的高 1。`nooffset` 与 `offset 0` 同义。

`trianglepattern <bitpattern>` , `<bitpattern>` 必须是介于 0 和 7 之间的数 (3 个 bit)，被解释为位模式。每个 bit 决定了一个三角曲面¹子三角形的一边的可见性。bit 0 表示水平边。bit 1 表示竖直边。bit 2 表示对角边 (分割网格为 2 个 3 角形的"对角线")。默认值为 3，这意味着水平和竖直线可见，对角线不可见。

1 三角曲面，一个曲面在计算机表示时被剖分为若干个 3 角形，3 角形数目越多曲面越精确平滑。



undefined <level> 让你决定在发现未定义值时算法的行为（missing data 或者 函数无意义的取值或者返回值），超出 x y z 轴区间以外。这些无效点被丢弃，或者按照指定方法绘制。曲面上接近一个已被丢弃的点的其他点也被丢弃。这将在曲面上产生一个洞。<level>=3 等于使用了 noundefined 关键字，没有任何点被丢弃。这可能产生各种错误，避免这样。<level>=2 抛弃未定义值，但是保留超出区间的值。<level>=1 表示 抛弃在区间之外的值。

如果 undefined 关键字出现，noaltdiagonal 关键字，可以设置发现特殊情况时的处理办法。曲面的每个网格被对角线分割为两个三角形。通常，所有的对角线相对与网格拥有相同的方向。出现未定义值，网格 4 个角之一会被丢弃。如果这个角处在对角线上两个三角形都会被丢弃。使用默认值 altdiagonal

可以让 gnuplot 使用另一个对角线分割网格，使得图形的缺损最小。

bentover 选项控制当其他情况发生时所做的处理。此选项和 trianglepattern 连用。对于褶皱曲面，可能会发生一个网格按照算法从正面被分割但是从反方向观察。

original quadrangle:	A--B	displayed quadrangle:	C----B
("set view 0,0")	/	("set view 75,75" perhaps)	\
	/		\
	C--D		\
			A D

如果 trianglepattern 没有显示对角线，那么产生的图形可能难以理解。这种情况下 bentover 自动显示对角线。如果你不希望这样请使用 nobentover 选项。

参见实例：hidden.dem sigulr.dem

Historysize

set historysize 只在 gnupot 编译时使用了 gnu readline 库才有效。

语法：

```
set historysize <int>
unset historysize
```

开启历史记录，<int>默认值为 500. 超出<int>的历史被丢弃。unset historysize 使得历史记录无限，并且保存进历史文件。

Isosamples

控制函数曲面绘制的等值线密度。可以使用 set isosamples 命令修改。

语法：

```
set isosamples <iso_1> {,<iso_2>}
show isosamples
```

每个函数曲面都有 <iso_1>条 u 轴 等值线和 <iso_2>条 v 轴 等值线。如果只设置了 <iso_1>，那么 <iso_2> 默认等于<iso_1> 默认采样设置为 10.高的采样值，获得更高的曲面精度，但是时间更长。

这些参数在数据文件绘图时无效。

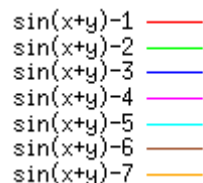
一条等值线是曲面函数一个参数固定同时另一个参数变化所得到的曲线。函数 $s(u,v)$ 的<iso_1> 条 u 轴等值线，意为：产生<iso_1> 条， $c(v)=s(u_0,v)$ 曲线。与之类似 v 轴等值线意为： $c(u)=s(u,v_0)$

如果没有开启 hidden 选项，那么 set samples 设置在每根等值线上的采样。等高线算法依赖于函数在等值线上采样，所以 改变 samples 会引起曲面和等高线的变化。

Key

set key 选项为数据指定标题和样例格式。

图例的内容可以是：数据集中的命名字段和函数和采样线或者用于表示它们的符号，由 title 和 with 关键字设置。图例默认出现在图形右上角



```
sin(x+y)-1
sin(x+y)-2
sin(x+y)-3
sin(x+y)-4
sin(x+y)-5
sin(x+y)-6
sin(x+y)-7
```

如上图，左边称作样例标题，右边称作曲线样例。

语法：

```
set key {on|off} {default}
      {{inside | outside} | {lmargin | rmargin | tmargin | bmargin}
       | {at <position>}}
{left | right | center} {top | bottom | center}
{vertical | horizontal} {Left | Right}
{{no}opaque}
{{no}reverse} {{no}invert}
{samples <sample_length>} {spacing <vertical_spacing>}
{width <width_increment>}
{height <height_increment>}
{{no}autotitle {columnheader}}
{title "<text>"} {{no}enhanced}
{font "<face>,<size>"} {textcolor <colourspec>}
{{no}box { {linestyle | ls <line_style>}
           | {linetype | lt <line_type>}}
{linewidth | lw <line_width>}}
{maxcols {<max no. of columns> | auto}}
{maxrows {<max no. of rows> | auto}}

unset key
show key
```

对于每次绘制，图例包含标题和样例(线、点、box)。图例可以被 set key off、unset key 关闭。单个图例可以被 plot 语句的 notitle 关键字关闭。

各个图例之间根据 vertical 或者 horizontal 关键字相互堆叠。vertical 关键字使得图例竖直堆叠。maxrows 选项控制竖直堆叠的最大行数。Horizontal 关键字使得图例水平堆叠。这两个选项在 outside 模式下有效。

关键字 left right top bottom center inside outside lmargin rmargin tmargin bmargin 可

以用于表示图例放置的位置例如：left bottom 放置在左下角，right bottom 右下角。bmargin 放置在底边空白处。使用 set key at 控制图例出现坐标，如果坐标在可见区域外可能导致看不到图例。

Left 和 Right¹ 控制图例标题的对齐设置。reverse 倒转文本和样例。box 关键字在图例周围加上边框，边框的线条样式可以用 lt lc lw 等关键字设置。

默认情况下图例在绘制时产生。图例标题和样例与曲线绘制同时产生。这意味着新的绘制可能会把图例放置到旧图例前方。Set key opaque 使得所有图例在最后统一绘制。默认情况下第一次绘制的图例在最上面，随后的图例向下堆叠。invert 关键字使得第一个图例在最下面，随后的图例向上堆叠。

samplen 控制样例的宽度(控制曲线样例的长度)。spacing 单个样例的竖直高度(控制行间距，值过小文字会重叠)。width 控制边框离文字边沿的水平距离。height 控制图例方框的高度相对与字符高度的增减值。width 和 height 一起控制边框的大小。这个选项在你希望图例方框比图例更大时有用。

图例默认标题由 autotitles 选项自动产生，但是 plot 可以使用 title 关键字明确设置图例标题。

set key autotitle columnheader 使得自动为每个输入字段生成字段名，参见：using 80。你可以使用 title columnhead(3)，使用第 3 字段的字段名作为标题。如果绘制使用了内联函数并且它引用了多个字段，那么 gnuplot 将无法确定具体使用哪个字段的字段名作为标题，你需要使用 title columnhead() 来明确设置标题。

```
plot "data" using (($2+$3)/$4) title columnhead(3) with lines
```

set key title "" 可以在样例列表的顶部放置一个图例标题。图例标题使用与样例标题相同的对齐设定。输入 set key default 使 key 设置恢复到默认状态。

其默认值为：








```
set key on right top vertical Right noreverse \
noinvert samplen 4 spacing 1.25 title "" nobox
```

图例在每次 plot 时自动产生。在图例标题右边绘制曲线样例,如果使用了 reverse 关键字 样例出现在标题左边。








使用 tex 等能在字符串内包含格式信息的输出终端，gnuplot 只能估计字符串本身的长度。所以有时候出现问题，你可能需要使用 set key left Left reverse 使得这个问题不那么明显。

splot 可以绘制等高线，等高线的样例也会出现在图例内。如果对齐效果不好。你需要搭配 set clabel 选项。设置等高线的标题格式。








1 注意大小写：Left 设置标题左对齐，left 控制图例放在绘图区的左边。

`title ""`
`AMAA BA1` 
`A2` 
`AA A3` 
`A nBA4` 
`AA AA5` 
`AA6` 
`AA AAA7` 








`set key title nobox` , `set key notitle nobox`

`AMAA BA1` 
`A2` 
`AA A3` 
`A nBA4` 
`AA AA5` 
`AA6` 
`AA AAA7` 








`set key box 3`

box 3	
AMAA BA1	
A2	
AA A3	
A nBA4	
AA AA5	
AA6	
AA AAA7	




`set key left box 3`

Left	
AMAA BA1	
A2	
AA A3	
A nBA4	
AA AA5	
AA6	
AA AAA7	

`set key Right box 3`

Right	
AMAA BA1	
A2	
AA A3	
A nBA4	
AA AA5	
AA6	
AA AAA7	

`set key maxrow 3 box 3`

maxrow 3		
AMAA BA1		AA AAA7
A2		AA6
AA A3		

实例：

恢复图例设置为默认值。

```
set key default
```

关闭图例：

```
unset key
```

设置图例出现在第一坐标系统 2,3.5,2 位置：

```
set key at 2, 3.5, 2
```

设置图例出现在图形下方：

```
set key below
```

设置图例出现在图形左下角，设置标题为 Legend，以线条样式 3 绘制边框。

```
set key left bottom Left title ' Legend' box 3
```

图例位置：

理解位置选项的含义，你需要理解屏幕区域的划分。讨论简写：

left/center/right == l/c/r

top/center/bottom == t/c/b

当处在 inside 模式下，图例放置在绘图区域以内。各个选项的含义是：

t/l	t/c	t/r
c/l	c	c/r
b/l	b/c	b/r

处在 outside 模式下，图例放置在绘图区域到画布边缘之间。当然这需要向内调整绘图区边界，为图例留出空间。在某些终端下会发生错误。outside 模式下，绘图区域边界的变化受到图例放置位置和堆叠选项影响。如果图例在 outside 模式下被放置在角落上：vertical 使得左或右边界会向内变化。

Horizontal 导致顶或底边界向内变化。概念类似这样：

outside top horizontal == tmargin

outside bottom horizontal == bmargin

outside left vertical == lmargin

outside right vertical == rmargin

各个选项含义为：

	l/tm	c/tm	r/tm	
t/lm				t/rm
c/lm				c/rm
b/lm				b/rm
	l/bm	c/bm	r/bm	

关键字 above 和 over 是 tmargin 的同义词。如果只应用了 above over 默认使用 center horizontal 选项。below 和 under 是 bmargin 的同义词。如果只应用了 below 或 under 选项默认使用 center horizontal 选项。如果只设定了 outside 关键字，默认位置为：top right vertical。

实例：

请绘制 6 跟曲线看以下命令的效果。

```
set key bmargin
set key bmargin maxrow 2
set key tmargin
```

```
set key tmargin maxrow 2 box
set key rmargin box 2 title 'rmargin'
set key inside left top
set key outside left top hori
set key outside left top vertical
```

图例样本

默认情况下，每一个图形会自动产生合适的图例。其中包含了图形标题和图形样例。字体和文字颜色由 plot 命令的 title 选项控制。Plot variable 关键字控制文本颜色：颜色指定 19。设置标签为样式 3 蓝色你需要这样：plot 'dat' using 1:2:3 with labels tc lt 3

图例样例的长度由 samplen 选项控制。样例长度由刻度长度加上 <sample_length> 乘以字符宽度。samplen 也会影响点样例的位置，因为点样例默认是在样例区居中。

行间垂直距离由 spacing 控制。Gnuplot 保证数值距离不会小于字符高度。

<width_increment> 单位为字符宽度，他作为字符串长度的增量或减量。这用于你在图例上使用了 box，并且字符串里使用了控制字符。gnuplot 简单的统计字符串内字符数目，然后计算盒子宽度。这个选项可以纠正这个错误。

Label

在任意的位置放置标签

语法:

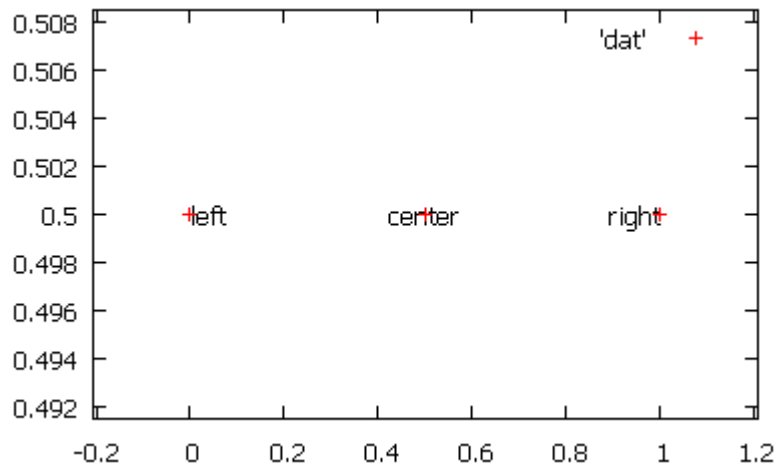
```
set label {<tag>} {"<label text>"} {at <position>}
    {left | center | right}
    {norotate | rotate {by <degrees>}}
    {font "<name>{,<size>}" }
    {noenhanced}
    {front | back}
    {textcolor <colourspec>}
    {point <pointstyle> | nopoint}
    {offset <offset>}
unset label {<tag>}
show label
```

<position> 为坐标，参见：坐标 6。

<tag> 为一个整数，他代表标签的 ID。如果没有设置 <tag> 会自动分配一个最小的可用 ID。ID 用于时候更改指定标签的属性，或者删除标签。

<label text> 可以使变量或字符串常量或者函数返回值。默认文本的左端与坐标对齐。使用关键字：

right 靠右对齐 center 文本居中 left 靠左对齐。



如果只出现 rotate 关键字标签被垂直放置。使用 rotate by <degrees> 使得标签相对于水平位置旋转 30 度。

font "<name>,<size>" 用来明确指定字体和大小。

通常如果开启了增强文本模式所有的字符串都会作为增强文本处理。使用 noenhanced 关键字，对某条标签关闭增强模式。

Front 关键字，使得标签在图形上方（可以遮蔽图形和曲线）。Back 使得标签在图形后方（可以被图形遮蔽）。如果你绘制了密集的点，front 可以避免标签难以认清。

Textcolor <colourspec> 设置某标签的文字色彩。Colourspec 可以是 线条类型，RGB 色彩，调色板映射。

```
`tc default` 设置色彩为默认值
`tc lt <n>` 设置色彩为线条类型<n> 的颜色
`tc ls <n>` 设置色彩为线条样式 <n>.的颜色
`tc palette z` 根据标签Z值，得到调色板色彩。
`tc palette cb <val>` 根据<val> 得到 colorbar.上的对应色彩。
`tc palette fraction <val>`， 0<=val<=1, 根据调色板的灰度和色彩求得val对应的色彩。
`tc rgb "#RRGGBB"` 24-bit RGB 色彩
```

如果使用<pointstyle> 参数，可以使用关键字 lt pt ps 更改点的样式,由pt设置点类型，ps设置点大小，lt 设置色彩。：

```
set label "hello" at 0,0 point pt 8 lt 8
```

如果使用了 point关键字。那么可以使用 offset 关键字他设置点到标签的相对位置。用一个向量表示。向量是坐标表达式。

如果某个坐标轴是时间序列。那么坐标值可以是有效的时间字符串。

EEPIC Imagen Latex TPIC 终端驱动可以使用 \\ 表示换行。

使用labels 样式可以从文件中读取标签坐标和文本。参见：49。

实例：

设置标签"y=x" 在坐标(1,2)：

```
set label "y=x" at 1,2
```

设置标签S，使用symbol字体，24字号。并且相对于绘图区居中：

```
set label "S" at graph 0.5,0.5 center font "Symbol,24"
```

设置标签 "y=x^2" 在 (2,3,4)坐标处，并且右对齐。标签ID为 3：

```
set label 3 "y=x^2" at 2,3,4 right
```

修改ID=3 的标签居中对齐。：

```
set label 3 center
```

删除标签2

```
unset label 2
```

删除所有标签：

```
unset label
```

按照ID顺序显示所有标签：

```
show label
```

设置标签坐标为时间字符串：

```
set timefmt "%d/%m/%y,%H:%M"  
set label "Harvest" at "25/8/93",1
```

To display a freshly tted parameter on the plot with the data and the tted function, do this after the t, but before the plot:

```
set label sprintf("a = %3.5g", par_a) at 30,15  
bfit = gprintf("b = %s*10^%S", par_b)  
set label bfit at 30,20
```

To display a function denition along with its tted parameters, use:

```
f(x)=a+b*x  
fit f(x) 'datafile' via a,b  
set label GPFUN_f at graph .05,.95  
set label sprintf("a = %g", a) at graph .05,.90  
set label sprintf("b = %g", b) at graph .05,.85
```

设置标签相对于坐标的偏移：

```
set label 'origin' at 0,0 point lt 1 pt 2 ps 3 offset 1,-1
```

设置标签的色彩为根据Z值获取调色板色彩(pm3d 的色彩标尺)。

```
set label 'text' at 0,0,5.5 tc palette z
```

linetype

Set linetype 命令允许你重新定义基本 linetype。命令选项与 set style line 命令一样。与线条样式不同，set linetype 对线条类型的改变不会被 reset 命令重置。

实例：

将 线条类型 1 和 2 设置成蓝色和绿色：

```
set linetype 1 lw 2 lc rgb "blue" pointtype 6
set linetype 2 lw 2 lc rgb "forest-green" pointtype 8
```

1被设置为粗的蓝色线条，而不是默认红色细线。

注意：这个命令式gnuplot 4.6 新增的。它用于取代4.2版本的"set style increment user" 命令。旧的命令已经抛弃。

这项机制可以用于定义用户自己的样式列表。建议把这件事情放到初始化脚本~/.gnuplot中完成。类似这样：

```
if ((GPVAL_VERSION < 4.5) \
|| (!strstrt(GPVAL_COMPILE_OPTIONS, "+USER_LINETYPES"))) \
exit
set linetype 1 lc rgb "dark-violet" lw 2 pt 0
set linetype 2 lc rgb "sea-green" lw 2 pt 7
set linetype 3 lc rgb "cyan" lw 2 pt 6 pi -1
set linetype 4 lc rgb "dark-red" lw 2 pt 5 pi -1
set linetype 5 lc rgb "blue" lw 2 pt 8
set linetype 6 lc rgb "dark-orange" lw 2 pt 3
set linetype 7 lc rgb "black" lw 2 pt 11
set linetype 8 lc rgb "goldenrod" lw 2
set linetype cycle 8
```

每当你运行 gnuplot 时，初始化脚本会自动定义这些线条类型。最开始的 if 用来判断当前 gnuplot 是否支持 set linetype。

类似的脚本可以设置绘制的"主题"。Set linetype cycle 8 命令告诉 gnuplot 当 plot 使用的 linetype 高于 8 时从头开始循环也就是： $\text{linetype}(N) = (N-1) \% 8 + 1$ 。点的属性不受这个命令影响。Unset linetype cycle 关闭 linetype 循环。

Lmargin

Set Lmargin 控制图形区和画布区的左边界。参阅 set margin

loadpath

Loadpath 选项设置着 call load plot splot 命令对数据或脚本的默认搜索路径。如果指定文件不在工作目录下那么搜索 loadpath 路径。

语法：

```
set loadpath {"pathlist1" {"pathlist2"...}}
show loadpath
```

路径是一个字符串。可以是单个路径。或由操作系统支持的路径分隔符分割的路径列表。Dos/win/OS 平台使用分号，Unix/linux 使用冒号。

Locale

Set locale 设置本地选项。它影响默认的坐标轴数字、时间和日期显示格式。

语法：

```
set locale {"<locale>"}
```

<locale> 为你系统中安装了的本地设置。具体需要查阅你的操作系统手册。使用 set locale "" 将尝试根据环境变量 LC_TIME LANG LC_ALL 自动推断。

改变小数点符号参阅 set decimalsign 106. 改变字符编码使用 set encoding 108。

logscale

语法：

```
set logscale <axes> {<base>}  
unset logscale <axes>  
show logscale
```

设置坐标轴为对数坐标。<axes> 可以是 x x2 y y2 z cb 和 r，他们以任何顺序出现。<base> 是对数的底，默认为 10.命令 unset logscale 对所有坐标轴关闭对数。

实例：

```
为 x and z 轴开启对数坐标：  
set logscale xz  
为 y 轴开启以 2 为底的对数坐标：  
set logscale y 2  
为 pm3d 样式的 z 和 颜色标尺开启对数：  
set logscale zcb  
关闭 z 轴的对数坐标  
unset logscale z
```

macros

如果命令行宏扩展功能打开。那么在命令行中的 @<stringvariablename> 格式会被扩展。stringvariablename 是一个字符串变量名。宏会被扩展为字符串值。参阅：变量替换和宏 25 26

语法：

```
set macros
```

mapping

如果你数据文件中的坐标是球面坐标或柱面坐标。使用 set mapping 命令可以让 gnuplot 自动转化他们。

语法：

```
set mapping {cartesical | spherical |cylindrical }
```

默认使用笛卡尔坐标系。对于球面坐标系统，数据包含了 2 列或 3 列。前两列会被解释为方位角 theta 和极坐标偏角 phi。如果有第三列那么第三列作为半径。否则自动设置半径值为 1。角度单位由 set angles 控制。

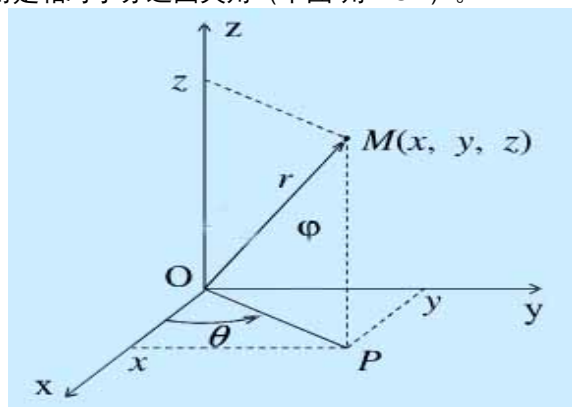
映射公式：

$$x = r * \cos(\theta) * \cos(\phi)$$

$$y = r * \sin(\theta) * \cos(\phi)$$

$$z = r * \sin(\phi)$$

注意这是地理球面系统。Phi角是相对于赤道面夹角（下图 角MOP）。



对于柱面坐标系统，数据可以包括2列或3列。头两列被解释为theta和Z，半径由第三列决定，如果没有第三列则为1。

映射公式为：

$$x = r * \cos(\theta)$$

$$y = r * \sin(\theta)$$

$$z = z$$

当然简单的数据，你可以自己推倒映射函数，然后写在using。

例如柱面映射：

```
using ($3 * cos($1)):($3*sin($1)):$2
```

但是set mapping 会给你提供方便。Mapping选项不会影响2d绘制。

参考实例：world.dem

Margin

margin 控制绘图区和画布边缘的间距。margin 有默认设置。一般来说默认值效果不错。不过使用 set margin 你可以修改页边距。Show margin 显示当前边距设置。

语法：

```
set bmargin {{at screen} <margin>}
set lmargin {{at screen} <margin>}
set rmargin {{at screen} <margin>}
set tmargin {{at screen} <margin>}
show margin
```

<margin>默认单位为字符高度或者宽度。正数定义了一个绝对页边距。省略或者负数值导致 gnuplot 自动计算页边距。对于 3d 绘制，只有左边距可以字符为单位。

At screen 关键字表示<margin>为页边距占画布的比例。

通常页边距根据页边距内容自动计算：刻度 和坐标轴标签、图例。刻度和刻度标签的占位都不被自动计算。所以如果页边距出现了其他文字同时 margin 设置的很小。那么可能会出现文字重叠。

Mouse

Set mouse 对于交互终端开启鼠标功能。它通常在交互终端下默认打开。但是如果命令从文件中载入不会自动开启。

有两个鼠标模式。2D 模式工作于 plot 和 splot 命令 (set view z 0 90 180)。在这个模式下鼠标被追踪。你可以移动或者缩放。某些终端支持单击图形样例，对某个图形独立的关闭或者开启。

对于 3d 图形，splot：使用鼠标键 1 控制视角 和鼠标键 2 用来缩放。

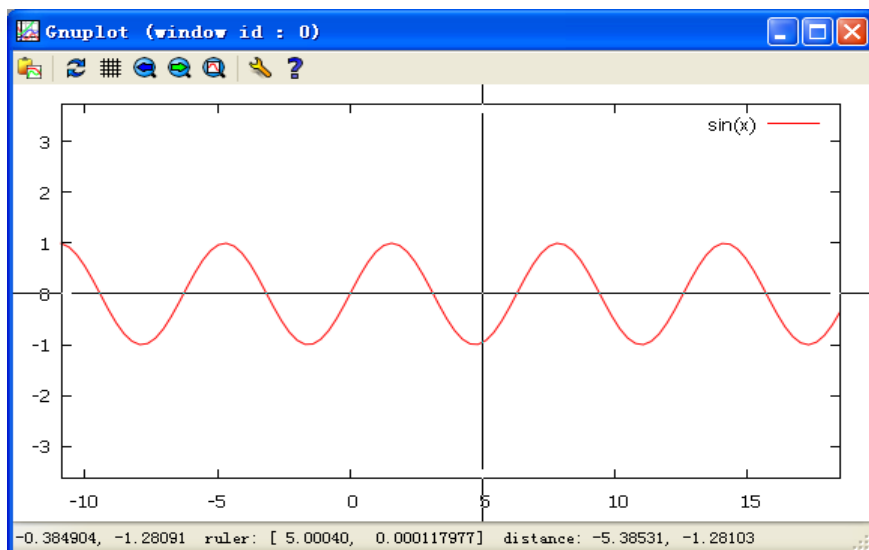
- 拖动鼠标左键会旋转视角，ctrl+拖动鼠标左键旋转视角时避免绘制图形，直到鼠标松开（数据量很大时，即时绘制会很卡）。
- 按住 ctrl+ 滚轮上下会缩放图形，xrange 和 yrange 也随之变化。
- 只有滚轮上下会平移 xrange。shift+滚轮上下平移 yrange。
- 拖动鼠标中键改变 zrange 的显示比例。ctrl+拖动鼠标中键改变 zrange 的显示比例避免绘制图形，直到鼠标松开。

鼠标在 multiplot 模式下不可用。使用了 unset multiplot 时鼠标再次被打开。

语法：

```
set mouse {doubleclick <ms>} {nodoubleclick} \  
  {{no}zoomcoordinates} \  
  {noruler | ruler {at x,y}} \  
  {polar distance{deg|tan} | nopolar distance} \  
  {format <string>} \  
  {clipboardformat <int>/<string>} \  
  {mouseformat <int>/<string>} \  
  {{no}labels {"labeloptions"}} \  
  {{no}zoomjump} {{no}verbose}  
unset mouse
```

noruler 和 ruler 会关闭或者开启标尺。At x,y 设置标尺原点在 x,y。当标尺开启，交互终端上会连续的显示鼠标相对于标尺的距离。



上图 标尺设置在 5,0 终端下方状态栏显示了标尺相关信息。如果使用 noruler 状态栏只有：

同时图形上也不会出现标尺。在交互终端下，字符 `r` 的绑定用来随时对当前鼠标位置开启和关闭标尺。

选项 `polardistance` 决定鼠标和标尺之间的关系是否使用极坐标表示。切换此选项被绑定到数字键 5。

`Labels` 关键字当单击鼠标键 2 会在单击位置永久显示标签。`noLabels` 关键字当单击鼠标键 2 会在单击位置临时显示标签。标签的内容由 `mouseformat` 设置。`labeloptions` 字符串被传递给 `set label` 命令。默认是 `"point pointstyle 1"`，它会在标签位置绘制一个加号。临时标签在下次刷新后消失。永久标签的关闭可以通过在标签位置使用 `:Ctrl+鼠标 2` 完成。标签位置上的符号 `pointsize` 决定有效单击区域。

`verbose` 关键字使得鼠标事件产生的命令在执行阶段显示。数字键 6 的绑定用于切换 `verbose` 选项。默认时 `verbose` 处在关闭状态。

在交互窗口上按下 `h`，会在命令行标准输出显示简短的功能键介绍。它包括用户定义的绑定和内置的绑定。参考 `bind 21`。

doubleclick

双击事件是在指定时间间隔（默认 300ms）内连续两次单击鼠标 1。它会复制当前鼠标位置到系统剪切板¹。如果时间间隔设置为 0，那么每次单击鼠标便会复制。

Mouseformat

命令 `set mouse format` 设置一个由 `sprintf` 函数处理的字符串，它决定了鼠标坐标如何显示和复制到剪切板，默认是 `"% #g"`。

`set mouse clipboardformat` 和 `set mouse mouseformat` 分别设置即时坐标显示格式和剪切板坐标格式。`Set mouse mouseformat ""`，关闭鼠标即时坐标显示。一个整数可以用来代表内定格式，见下表。

The following formats are available:

0	default (same as 1)	
1	axis coordinates	1.23, 2.45
2	graph coordinates (from 0 to 1)	/0.00, 1.00/
3	x = timefmt y = axis	[(as set by 'set timefmt'), 2.45]
4	x = date y = axis	[31. 12. 1999, 2.45]
5	x = time y = axis	[23:59, 2.45]
6	x = date time y = axis	[31. 12. 1999 23:59, 2.45]
7	format from 'set mouse mouseformat', e.g. "mouse x,y = 1.23,	2.450"

滚动

X 和 Y 坐标轴可以用鼠标滚轮校准。滚轮上 将 Ymax 和 Ymin 向正方向平移 `yrange` 的 10%。滚轮下向反方向平移。`shift+滚轮上` 向负方向平移 Xmin Xmax；`Shift+滚轮下` 向正方向平移。`ctrl+滚轮上` 向屏幕中心放大；`Ctrl+滚轮下` 缩小。`shift+ctrl+滚轮上下`，只在 X X2 轴缩放。

¹ linux 下你应该使用鼠标中键粘帖；

X11

如果使用 `set term X11 <n>` 开启了多个 x11 终端。那么只有当前绘制窗体可以接受完成的鼠标和键盘操作。其他窗体只在左下角显示鼠标坐标。

缩放

`ctrl`+滚轮上向屏幕中心放大；`Ctrl`+滚轮下缩小。`shift+ctrl`+滚轮上下，只在 X X2 轴缩放。缩放操作有历史记录。最初的绘制被保留下来，可以使用 `u` 键撤销所有缩放。`p` 撤销一个缩放，`n` 重做一个缩放。

单击鼠标 3 可以开启缩放框。它决定了缩放区域。

选项 `zoomcoordinates` 决定了缩放框的坐标是否显示在边缘。默认开启。

如果 `zoomjump` 选项开启，在开始缩放框时鼠标会自动跳跃一小段距离。这可以避免误操作导致极小的缩放框。`Zoomjump` 默认关闭。

Multiplot

`Set multiplot` 开启多绘制模式。在此模式下多个绘制被放到一个窗体或一个画布上。

语法：

```
set multiplot { layout <rows>,<cols>
                {rowsfirst|columnsfirst} {downwards|upwards}
                {title <page title>}
                {scale <xscale>{,<yscale>}} {offset <xoff>{,<yoff>}}
            }
unset multiplot
```

对于某些终端除非使用 `unset multiplot`，否则绘制不会输出。这条命令导致绘制输出，并且回到单绘制模式。其他的终端每使用一次 `plot` 命令，便产生一次输出（根据输出终端的不同，既可能是重绘所有的图形，也可能是只添加新的图形）。

下一次绘制所使用的区域不会自动清除。`Clear` 命令用于清除绘制区。

任何在 `plot` 之前定义的箭头和标签都会在每次绘制中出现。所以你应该在每次 `pot` 之前重新设置一遍箭头和标签。如果你希望某些元素只出现在一副图形中你应该在 `plot` 前 `set`，`plot` 后及时 `unset`。

`multiplot` 模式的标题与它所包含绘制的标题无关。标题所需要的空白区域会在页顶部保留，空白宽度等于页宽度。

`Set origin` 和 `set size` 用来控制每个绘制发生的区域，请参考她们各自的说明。

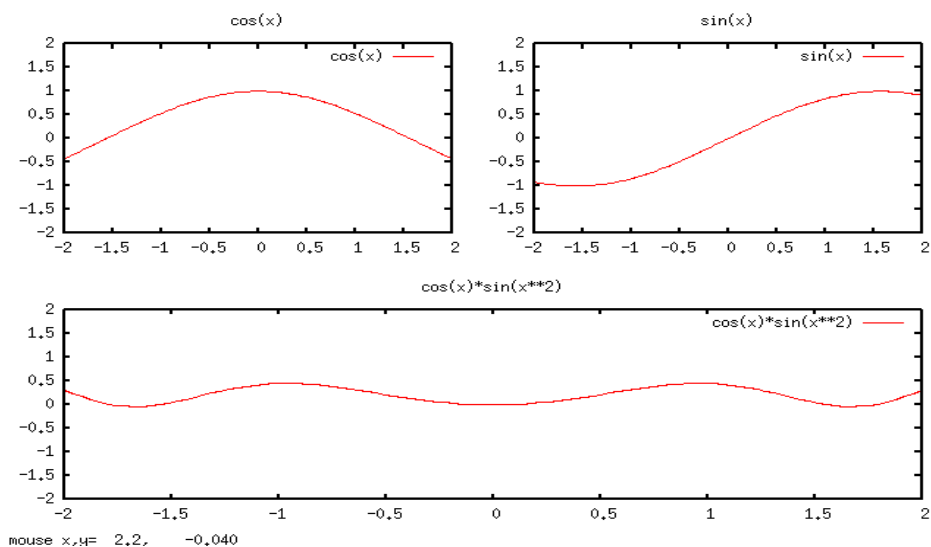
实例：

```
set multiplot
set size 1,0.5
```

```

set origin 0,0
set title "cos(x)*sin(x**2)"
plot cos(x)*sin(x**2)
set size 0.5,0.5
set origin 0,0.5
set title "cos(x)"
plot cos(x)
set size 0.5,0.5
set origin 0.5,0.5
set title "sin(x)"
plot sin(x)
unset multiplot

```



set size 和 set origin 代表整个页面的比例。Set term size 决定整个页面的大小。可以使用 set margins 设置页边空白。注意：页边设置是绝对值，它以字符大小为单位。所以对于不同输出终端如果它们使用不同字体，页边可能会不一样（例如：显示设备和打印机）。

layout 选项可以用来给多副图形生成简单的排版，从而避免反复的使用 set size 和 set origin。Layout 为每个 plot 自动 set size 和 set origin，当然如果你在 plot 之前设置了它们那么 layout 不会帮倒忙。layout 排版的格式就是简单的按照行列对齐。它会生成 <rows> 行和 <cols> 列的无边框排版。Downwards 或 upwards 决定堆叠方向。Rowsfirst 和 downwards 决定行填充优先还是列优先。默认是 rowsfirst 和 downwards

每个绘制可以被 scale 选项缩放，offset 选项平移。如果 scale 或者 offset 的 y 参数没有提供那么自动使用 x 参数值。Unset multiplot 会关闭自动排版，并且将 set size 和 set origin 恢复为 set multiplot layout 之前的值。

实例：

```

set size 1,1
set origin 0,0
set multiplot layout 3,2 columnsfirst scale 1.1,0.9

```

```
[ up to 6 plot commands here ]
unset multiplot
```

会导致 6 副图形从上到下然后从左到右排列。

参考实例：multiplt.dem

Mx2tics

Set mx2tics 用于控制 x2 轴副刻度。

Mxtics

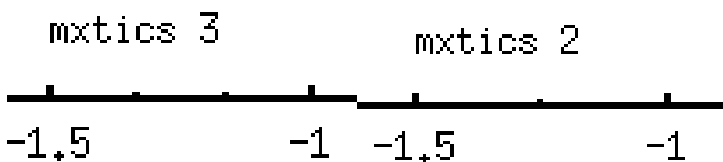
set mxtics 用于控制 x 轴副刻度。unset mxtics 关闭 x 轴副刻度。其他坐标轴的副刻度开启和关闭使用类似的语法。

语法：

```
set mxtics {<freq> | default}
unset mxtics
show mxtics
```

同样的语法可用于：mytics,mztics,mx2tics,my2tics,mcbtics

<freq>是主刻度之间副刻度的间隔个数（不是刻度个数）。



默认值对于线性坐标轴是 2 或 5. 其他情况是 1 或者 4. 使用 default 恢复为默认值。

如果坐标轴是对数坐标轴。<freq>会被设置为合适的值。

想在任意位置设置副刻度，使用 set {x|x2|y|y2|z}tics 命令并使得<label>为空，<level> 为 1 .

set m{x|x2|y|y2|z}tics 命令只在主刻度间距两两相等时有效。如果所有主刻度都是使用 set xtics 手动绘制的，副刻度命令被忽略。自动主刻度和手动主刻度可以混合使用，set xtics 和 set xtics add。

实例：

```
set xtics 0, 5, 10
set xtics add (7.5)
set mxtics 5
```

主刻度在 0,5,7.5,10，副刻度：1,2,3,4,6,7,8,9

```
set logscale y
```

```
set ytics format ""
set ytics 1e-6, 10, 1
set ytics add ("1" 1, ".1" 0.1, ".01" 0.01, "10^-3" 0.001, \
    "10^-4" 0.0001)
set mytics 10
```

主刻度使用指定格式，副刻度在对数位置。

副刻度，对于线性坐标轴是默认关闭的，对数坐标轴默认开启。它受 `axis|border` 和 `mirror` 的设置影响。参阅：`set xtics`

my2tics

控制 y2 的副刻度。参见 `set mxtics`

mytics

控制 y 的副刻度。参见 `set mxtics`

mztics

控制 z 的副刻度。参见 `set mxtics`

object

`set object` 命令定义一个将在 2d 绘制中出现的对象。你可以定义成任何你喜欢的东西。目前支持的对象包括：`rectangle`（矩形），`circle`（圆形），`ellipse`（椭圆型），`polygon`（多边形）。

这些对象的样式受到 `set style` 属性的影响。比如 `rectangle` 受到 `set style rectangle` 的影响，圆形、椭圆、多边形的填充受到 `set style fill` 的影响。不过每个对象可以单独设置样式属性。

语法：

```
set object <index>
    <object-type> <object-properties>
    {front|back|behind} {fc|fillcolor <colourspec>} {fs <fillstyle>}
    {default} {lw|linewidth <width>}
```

`<object-type>` 可以是：`rectangle`, `ellipse`, `circle`, `polygon`。每个类型有自己特殊的选项。

`front` 关键字 使得对象绘制在函数图形前方，但是处在拥有 `front` 属性的 `label` 后方。`Back` 关键字使得对象出现在所有图形最后。

```
set object rectangle from screen 0,0 to 1,1 behind
```

使得整个画布或页面拥有填充背景。`<colourspec>` 指定填充色，`fillcolor` 关键字可简写为 `fc`。填充样式使用 `<fillstyle>` 设置。如果关键字 `default` 出现那么使用默认样式和填充。

Rectangel

语法：

```
set object <index> rectangle
    {from <position> {to|rto} <position> |
    center <position> size <w>,<h> |
    at <position> size <w>,<h>}
```

矩形可以使用两种方式指定:1. 指定两个对角的坐标(from <pos> to <pos>)。2. 指定矩形中心和宽高(at <pos> size <w><h>)。这两种方法的参数是合法的坐标格式,可以是绝对坐标,也可以是相对与画布或者绘制区域的坐标。参见:坐标 6。关键字 at 和 center 等价。

实例：

为绘制区域绘制边框和背景。注意 back 关键字：

```
set object 1 rect from graph 0, graph 0 to graph 1, graph 1 back
set object 1 rect fc rgb "cyan" fillstyle solid 1.0
```

绘制从 0,0 到 2,3 的红色矩形：

```
set object 2 rect from 0,0 to 2,3 fc lt 1
```

绘制蓝色矩形边框（无填充矩形）：

```
set object 3 rect from 0,0 to 2,3 fs empty border rgb "blue"
```

设置对象 2 的填充和色彩为默认值：

```
set object 2 rect default
```

ellips

语法：

```
set object <index> ellipse {at|center} <position> size <w>,<h>
    {angle <orientation>} {units xy|xx|yy}
    {<other-object-properties>}
```

椭圆的位置通过 at <pos> size <w>,<h> 设置圆心和宽高（主轴和次轴）。主轴和次轴的长度必须在坐标系坐标,不可以是画布或者绘制区坐标。angle <orientation> 设置旋转角度,角度值等于主轴和 x 轴的夹角。units 关键字设置椭圆轴的缩放。units xy 意味着主轴长度对应 x 轴,次轴长度对应 y 轴（如果 x 轴和 y 轴比例尺¹不同,那么即使主轴值和次轴值相等看起来也不会是正圆）。unit xx 意味着主轴长度和次轴长度都根据 x 轴绘制。unit yy 与 unit xx 类似。默认是 xy,如果 set style ellipse 有 unit 设置那么使用此设置。

注意：如果 x 和 y 的比例尺不相等,那么旋转过后主轴和次轴的比例将不再正确。

set object ellipse size <2r>,<2r> 和 set object circle <r> 多数情况下会产生不一样的结果。圆

1 比例尺：x 轴可能用 1cm 长度代表数值 1, y 轴则可呢用 1cm 长度代表数值 0.5。这样(0,0)到(1,1)之间就不是正方形。

的半径长度由 x 轴比例尺决定。而 ellipse 的主轴和次轴默认时分别根据 x 轴和 y 轴比例尺绘制。如果 x 轴和 y 轴比例尺不同，那么即使主轴值和次轴值相等看起来也不会是正圆。使用这个，set size ratio -1 1,1，使得 x y 轴拥有相同单位长度。

Circle

语法：

```
set object <index> circle {at|center} <position> size <radius>
    {arc [<begin>:<end>]}
    {<other-object-properties>}
```

圆由圆心坐标和半径决定。圆心坐标和半径可以使用任何有效的坐标格式。半径的绘制总是依据水平轴比例尺，当然也可以是画布和绘图区坐标。任何水平和竖直缩放之后半径都会重新校准因此你总是能看到圆形。

默认时绘制一个完整圆。如果你只想要一个扇形，请使用 arc 选项设置起始角度和终止角度,角度是与 x 轴正方向的夹角。arc 始终是顺时针绘制[0:45]和[45:0]产生两个互补的"扇形"（区间两边别忘了方括号）。

Polygon

语法：

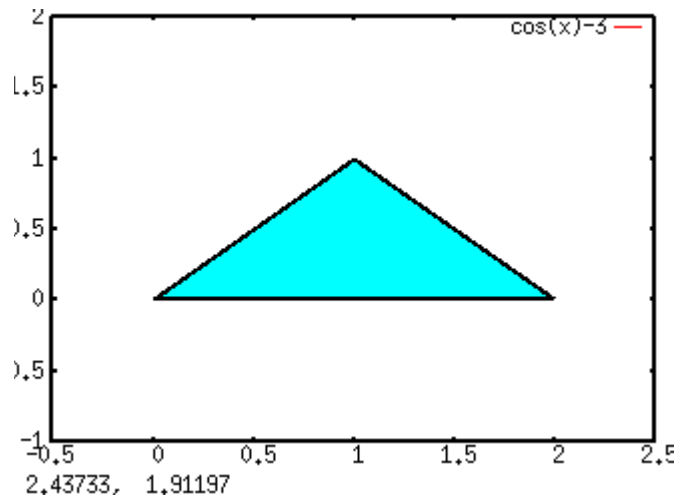
```
set object <index> polygon
    from <position> to <position> ... {to <position>}
or
    from <position> rto <position> ... {rto <position>}
```

多边形可以通过给定一系列顶点来绘制。可以使用任何有效的坐标格式。如果使用 rto 相对坐标那么坐标格式必须和第一个坐标一致。

实例：

```
set object 1 polygon from 0,0 to 1,1 to 2,0
set object 1 fc rgb "cyan" fs solid 1.0 border lt -1
```

产生一个三角形：



offsets

偏移机制允许在自动缩放机制下扩展值域。自动缩放会使得部分点临界在图像边沿。使用 offsets 让图像值域扩展，避免临界现象。偏移只影响 x1 和 y1 轴。并且只用于 2d plot 命令。

语法：

```
set offsets <left>, <right>, <top>, <bottom>
unset offsets
show offsets
```

偏移可以是常量或者是表达式。默认值为 0。left 和 right 偏移单位默认为 x1 轴单位。top 和 bottom 默认单位为 y1。使用 graph 关键字指定绘图区坐标。

正数会在指定方向上扩展 xrange 或者 yrange。例如 <bottom> 设置为整数那么 ymin 会更负。负数偏移也是可以的不过这会引起不期望的变化。使用 set auto fix，在校准值域时阻止自动缩放。

实例：

```
set auto fix
set offsets graph 0.05, 0, 2, 2
plot sin(x)
```

sin(x) 会被自动缩放到 [-1:1]，set offsets 使得上下延长 2 个单位于是变成了：[-3:3]。xrange 默认值是 [-10:10]，由于设置左偏移为 0.05 绘图区，因此扩展了 [-11:10]。

Origin

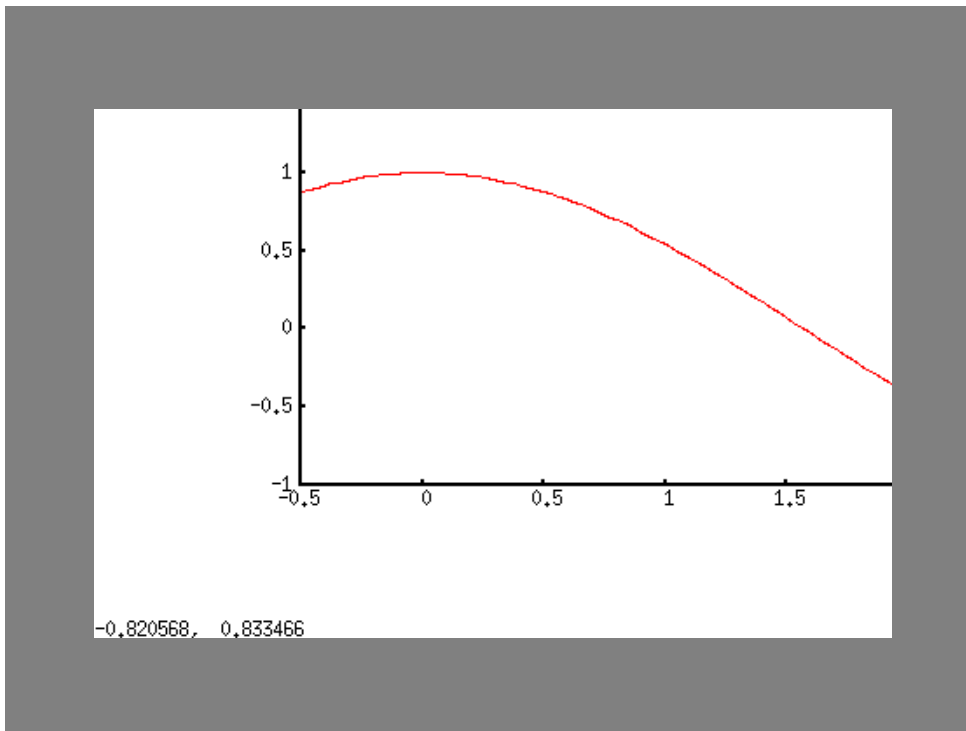
set origin 设置绘图原点。它的值是画布坐标。

语法：

```
set origin <x-origin>, <y-origin>
```

实例：

```
set origin 0.2, 0.2
plot cos(x)
```



output

默认值数据被输出到标准输出。set output 命令重定向输出到指定文件。

语法：

```
set output {"<filename>"}
show output
```

文件名必须以字符串形式出现。如果省略文件名，那么将文件输出到标准输出 STDOUT。

注意：set output "STDOUT" 可能会把文件输出到名为 STDOUT 的文件中。但是交互终端会忽略 output 设置。

交互终端会忽略 output 设置。当 set terminal 和 set output 都要设置时，最好把 terminal 先设置。因为某些终端需要为操作系统设置一些标志。某些文件操作系统在打开它之前需要预处理。

在支持管道的系统中，输出可以通过管道传递给命令。如果文件名的第一个非空字符为 |（管道符号）。那么使用管道命令。

例如：

```
set output "|lpr -Plaser filename"
set output "|lp -dlaser filename"
```

在 MSDOS 系统中，set output "PRN" 会自动定向到默认打印机。在 VMS，可以直接输出到 spooled device。也可以把数据输出到 DECnet transparent tasks。

Parametric

set parametric 使得 plot 或者 splot 可以理解函数的参数公式。unset parametric 关闭 parametric 模式。参考：Parametric 83 , dummy 108

语法：

```
set offsets <left>, <right>, <top>, <bottom>
unset offsets
show offsets
```

实例：

参数公式：

2d $x=f(t), y=g(t)$

3d $x=f(u,v), y=g(u,v), z=h(u,v)$

```
set parametric
plot f(t), g(t)
splot f(u, v), g(u, v), h(u, v)
```

对于 2d 绘制，参数公式需要一对函数。如果在 parametric 模式下 plot 没有足够的函数，那么会出错。

对于 3d 绘制，参数公式需要 3 个 2 元函数。

```
set parametric
splot cos(u)*cos(v), cos(u)*sin(v), sin(u)
```

绘制一个椭球。对于复杂的函数你可以使用自定义函数：

```
set parametric
fx(u, v)=cos(u)*cos(v)
fy(u, v)=cos(u)*sin(v),
fz(u)=sin(u)
splot fx(u, v), fy(u, v), fz(u)
```

实际上自变量的命名并不影响函数曲线。 $f(x)$ 和 $f(t)$ 可以绘制出相同的曲线，因为它们使用同一个函数。同理， $f(u,v)$ 和 $f(x,y)$ 是一个意思。自变量的命名参见：set dummy 108。

注意，参数公式之间使用逗号分割。如果不处在 parametric 模式：

```
splot fx(u, v), fy(u, v), fz(u)
```

将等同与：

```
splot fx(u, v)
replot fy(u, v)
replot fz(u)
```

当处在 parametric 模式下由于变量不再是 x y z , 所以 xrange yrange zrange 会更换为 trange urange vrange。所以 set xrange 应该改为 set trange。

Plot

show plot 命令显示最近一次的 plot/splot 命令 , 以及和它相关的 replot 命令。

show plot add2history 命令把当前 plot 命令添加到 history 历史里。这在你连续使用了 replot 时想编辑整个绘制命令时非常有用。

例如 :

```
splot f(x,y)
replot g(x,y)
```

会产生两条历史记录 , 当你使用方向键取回历史记录时只能获得两条独立的命令。但你可能想得到一个绘制两个函数的 splot 命令。使用 show plot add2history , 添加你想要的历史记录到历史列表:

```
splot f(x,y),g(x,y)
```

pm3d

pm3d 是 splot 的一个绘制样式 , 它拥有颜色标尺 , 可以用不同的颜色代表不同的值。它使用了一个可以不需要预处理就能把网格数据作为非网格数据绘制的算法。

语法 (可以以任何顺序出现) :

```
set pm3d {
    { at <position> }
    { interpolate <steps/points in scan, between scans> }
    { scansautomatic | scansforward | scansbackward | depthorder }
    { flush { begin | center | end } }
    { ftriangles | noftriangles }
    { cliplin | clip4in }
    { corners2color { mean|geomean|median|min|max|c1|c2|c3|c4 } }
    { hidden3d {<linestyle>} | nohidden3d }
    { implicit | explicit }
    { map }
}
show pm3d
unset pm3d
```

在全局绘制样式指定为 pm3d 时或者在 set pm3d implicit 时自动使用 pm3d 样式。在这两种情况下 pm3d 作为附加图形出现

在这两种情况下 :

```
splot 'fred.dat' with lines, 'lola.dat' with lines
```

实际上等同与 :

```
splot 'fred.dat' with lines, 'lola.dat' with lines
replot 'fred.dat' with pm3d, 'lola.dat' with pm3d
```

如果 explicit 开启或者 implicit 关闭那么必须明确使用 with pm3d 才能绘制 pm3d 平面。

这种情况下：

```
splot 'fred.dat' with lines, 'lola.dat' with pm3d
```

对于第一个文件只绘制了 lines，第二个文件只绘制了 pm3d 平面。

gnuplot 刚启动时处在 explicit 模式下。set pm3d;除了 implicit 选项，设置所有选项到默认值。

由于历史兼容问题，如果使用了 set pm3d (没有任何选项)或者 set pm3d at X (at 是第一个选项) 那么变更为 implicit 模式下。

如果你设置默认样式为 pm3d，例如：

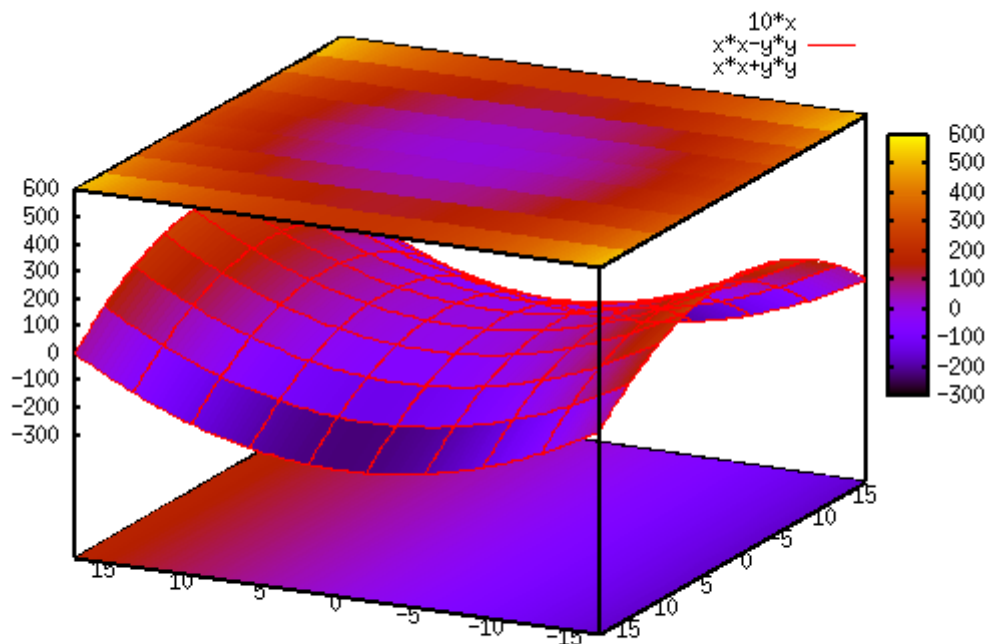
```
set style data pm3d
```

那么 implicit 和 explicit 无效。

如果你绘制了多个曲线，那么它们按照命令出现的顺序绘制。这在有填充的情况下会导致晚绘制的图像遮蔽与早先的图像重叠部分。

pm3d 色彩可以绘制在 3 个不同的位置：top、bottom、surface。参阅 pm3d position。以下命令在 3 个位置上绘制色彩。

```
set border 4095
set pm3d at s
splot 10*x with pm3d at b, x*x-y*y, x*x+y*y with pm3d at t
```



view: 65.0000, 207.000 scale: 1.00000, 1.00000

参考：set palette ,set cbrange ,set colorbox 102 。

参考实例：pm3d.dem

算法

我们先讨论映射或曲面是如何绘制的。数据来自于函数返回值或者是数据文件。每个曲面实际上由若干采样曲线构成。pm3d 算法填充由一次采样产生的相邻点到下一次采样获得的相邻点之间的区域，填充色由 4 个采样点相应的 z 值决定。默认填充色是 4 个 z 值的平均值。但是你可以使用 corners2color 选项改变它。为了得到更加合理的曲面。相邻的采样不会交叉，而且相邻采样的点数目也不会相差太多。当然，最理想的采样状态是每次都采集相同数目的点。另一个优势是，pm3d 算法不会在不存在数据的地方填充。

色彩填充：

1. splot 可以有 1 个或者 3 个输入列。色彩根据 4 个角的 z 值平均值决定。颜色映射发生在区间： $[\text{min_color_z}, \text{max_color_z}]$ ，或者是 cbrange 提供的灰度值 $[0:1]$ 。参见 set palette。
2. splot 有 2 个或者 4 个输入列。色彩映射根据最后一个输入列的值，而不是 z 值。这个可用于 4D 图像绘制。

其他注意事项：

1. 灰度和色彩根据值进行线性映射。标尺放置的位置使用 `set colorbox` 命令
2. 可以使用 `set view map` 将 `pm3d` 的 3d 曲面映射到 2d x y 平面这样你就得到一个特殊的 2d 图像。

position (位置)

色彩可以被填充到顶面和底面，或者是曲面本身。这使用 `at` 命令控制。`at` 的参数由 `b t s` 组合（最长 6 个字符）。例如，`at b` 在只底面绘制。`at st` 首先在曲面上填充然后在顶面填充。`at bstbst` 不是一个严谨的用法。

色块填充一个接着一个发生。当绘制在曲面上时，后产生的色块会覆盖先前的色块（`gnuplot` 不是一个计算交集的工具，它不会为重叠颜色产生“混合色彩”）。你也许要使用关键字 `scansforward` 和 `scansbackward` 强制采样顺序。默认值是 `scansautomatic`，此时 `gnuplot` 会自动猜测合适的顺序。`depthorder` 关键字为所有四边形根据深度重排序。色彩填充在排序之后发生。

参见 `pm3d depthorder`

scanorder (采样顺序)

默认 `pm3d` 是实心曲面，它们按照填充顺序相互遮蔽。这个顺序可以通过 `scansautomatic|scansforward|scansbackward` 三个关键字重新设置。这 3 个选项和 `set hidden3d` 并不完全兼容。

如果两次连续的采样点的数目不同，选项 `ftriangles` 设置了是否在曲面尾端自动填补三角形色块。它可用于绘制光滑的边沿。

`gnuplot` 并不真的从实心曲面中删除被遮蔽的部分。但是当有大量数据时，删除被遮蔽的部分对渲染速度有明显提升。这个模式使用以下选项开启：

```
set pm3d depthorder hidden3d
```

```
set hidden3d 并不影响 pm3d
```

clipping

剪取运算用于控制点在绘制区外时是否绘制。有 2 个选项，`clip1in`：四边形的四个点中至少有 1 个点在 `xrange` `yrange` 之内。`clip4in` 四边形 4 个点必须全部在 `xrange` `yrange` 内。

色彩赋值

3 个输入列 (x,y,z)

set palette 选项设置调色板映射。色彩根据调色板将数值映射成色彩。在每次绘制中只允许一个调色板。绘制多个平面使用多个调色板必须使用 multiplot 模式。为了防止色彩超出终端有效值，不要忘记使用 set palette maxcolors，它控制色彩上限。

色彩值影响 pm3d 四边形填充。这个色彩根据四边形四个角的 z 值计算。默认是取 4 个值平均值使用 corners2color <option> 改变这个算法。<option>可以为：mean(默认)、geomean、median 它产生不同种类的色彩。当<option> 为 min 和 max 选择最小或最大值。c1 c2 c3 c4 表示色彩计算只根据某个顶点的值。你可能要做些实验来确定 c1 对应哪个定点。由于 pm3d 算法不会自动扩展边沿的 3 角形为 4 变形。因此 c<j> 可能并不总是有对应点。举个例子在实例 pm3d.pm 中，pm3d 算法应用与 4x4 网格数据只产生了(4-1)*(4-1)=9 个色块

4 个输入列(x,y,z,color)

如果提供了第四个输入列那么它将被用作色彩映射。系统将不再根据色块四个定点 z 值来计算颜色。更加通用的情况是，使用 reg variable 可以使得色彩从最后输入列中读取。例如：

```
splot ... using 1:2:3:4 with pm3d lc rgb variable
```

线条颜色从第四输入列中读取。

其他的填充算法，可以使得某个四边形的填充色根据给定节点的 z 值（不是四边形顶点）。暂未实现。

Hidden3d

set pm3d hidden3d 命令为每个被绘制的四边形绘制边界线。通常它和 depthorder 选项连用，可以将被遮蔽的部分删除。这比 set hidden3d 更有意义。这个命令可以接受一个可选的颜色样式控制边界线的形态。如果颜色样式是负数或者省略，那么使用 plot 命令的对应属性。

例如：

```
set pm3d at s hidden3d depthorder
unset hidden3d
unset surf
splot x*x+y*y linecolor rgb "black" linewidth 0.5
```

采样补间

选项 interpolate m,n 将导致更精细的采样。对于数据文件，它影响色彩平面的平滑。对于函数合理设置采样补间会降低精度以节省内存。对于函数采样使用 set samples 和 set isosamples 更有意义。对于正数 m 或者 n，每个四边形在两个相互垂直的方向上被采样 m 次和 n 次。对于负数 m 和 n 它表示采样比率，它意味着最少 |m| 和 |n| 个点被绘制。interpolate 0,0 将导致自动选择。

废弃的选项

transparent solid 过于用于这个命令，但是现在你应该使用 `set grid {front|layerdefault}`。

`set pm3d map` 等价于

```
set pm3d at b ;
set view map;
set style data pm3d;
set style func pm3d;
```

提供它是为了反向兼容。

当使用数据文件时，小心使用 `zrange` 和 `cbrange`。参阅 `set (no)surface` 来查看可能的副作用。

Palette

`pm3d` 使用颜色标尺反应 z 值。调色板就决定了颜色和值的映射关系。它可用于填充等高线、矩阵图、色彩渐变的背景。

调色板的使用需要输出终端的支持，你可以使用 `set pm3d` 指令查看可用的终端类型。色彩范围不依赖于 `set cbrange` 和 `set log cb` 指令。`set colorbox` 控制色彩标尺的位置。

语法：

```
set palette
set palette {
    { gray | color }
    { gamma <gamma> }
    {
        rgbformulae <r>,<g>,<b>
        | defined { ( <gray1> <color1> {, <grayN> <colorN>}... ) }
        | file ' <filename>' {datafile-modifiers}
        | functions <R>,<G>,<B>
    }
    { cubehelix {start <val>} {cycles <val>} {saturation <val>} }
    { model { RGB | HSV | CMY | YIQ | XYZ } }
    { positive | negative }
    { nops_allcF | ps_allcF }
    { maxcolors <maxcolors> }
}
show palette
show palette palette <n> {{float | int}}
show palette gradient
show palette fit2rgbformulae
show palette rgbformulae
show colnames
```

set palette 不接受任何参数，会将 palette 复原到默认值。show palette 显示当前 palette 的设置。

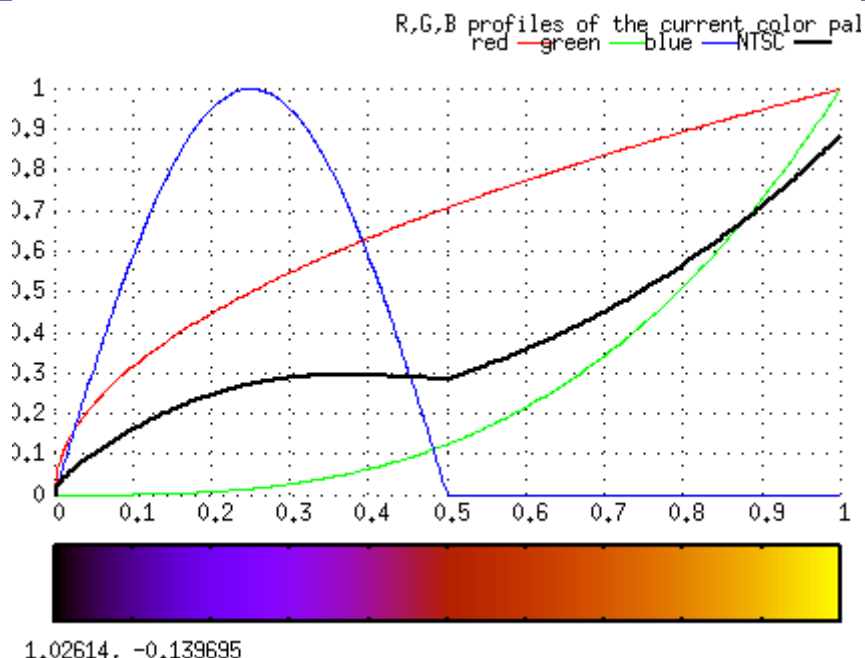
show palette gradient 显示调色板色彩剃度定义。show palette rgbformulae 显示灰度到色彩转换公式。show colnames 显示可用个的色彩名称（black read white 之类代指色彩值的单词）。

show palette palette <n> 显示一个 RGB 色彩表格，表格内 RGB 值根据当前的 palette 设置计算出 n 个不同的色彩，输出格式可以被 set print 选项更改。

```
gnuplot> show palette palette 10
Color palette with 10 discrete colors.
0. gray=0.0000, (r, g, b)=(0.0000, 0.0000, 0.0000), #000000 = 0 0 0
1. gray=0.1111, (r, g, b)=(0.3333, 0.0014, 0.6428), #5500a4 = 85 0 164
2. gray=0.2222, (r, g, b)=(0.4714, 0.0110, 0.9848), #7803fb = 120 3 251
3. gray=0.3333, (r, g, b)=(0.5774, 0.0370, 0.8660), #9309dd = 147 9 221
4. gray=0.4444, (r, g, b)=(0.6667, 0.0878, 0.3420), #aa1657 = 170 22 87
5. gray=0.5556, (r, g, b)=(0.7454, 0.1715, 0.0000), #be2c00 = 190 44 0
6. gray=0.6667, (r, g, b)=(0.8165, 0.2963, 0.0000), #d04c00 = 208 76 0
7. gray=0.7778, (r, g, b)=(0.8819, 0.4705, 0.0000), #e17800 = 225 120 0
8. gray=0.8889, (r, g, b)=(0.9428, 0.7023, 0.0000), #f0b300 = 240 179 0
9. gray=1.0000, (r, g, b)=(1.0000, 1.0000, 0.0000), #ffff00 = 255 255 0
```

通过这个方法，当前 gnuplot 的调色板可以加载到其他程序中，例如 octave。使用 test palette 可以查看当前调色板的色彩成分。

```
test palette
```



图像可以使用灰度或者彩色的方式引用调色板。例如，pm3d 色彩曲面中每个绘制点的灰度根据当前区块的四个顶点 z 值的平均值在 [min_z,max_z] 所占的比例得到 [0:1] 的灰度值。这个值可以直接用于黑白色彩绘制。或者根据 set palette rgbformulae 的色彩函数计算得到 RGB 彩色值，此函数把灰度 [0:1] 映射到 RGB [0:1] [0:1] [0:1]。

有两个映射类型：palette rgbformulae 和 palette functions

命令 show palette fit2rgbformulae 查询对于当前 palette 设置的最佳函数。

set palette gray 设置当前调色板为灰度调色板。它实际上设置了：set palette rgbformulae，使得色彩只映射到黑白色彩。palette functions、palette defined、palette rgbformulae、palette file 用于控制色彩映射。set palette color 自动设置一个合适的色彩映射。

自动伽玛校正由：set palette gamma <gamma> 设置。可用于灰度映射和 cubehelix 调色板。Gamma=1 那么产生密度的线性变化。参考：test palette。

多数终端只支持有限个数的色彩（gif 256 色），默认的线条色彩在 gnuplot 启动时就已经分配，剩余的色彩保留给 pm3d。对于使用多个不同调色板的 multiplot 绘制可能由于第一个调色板用掉了所有的色彩而失败。你应该使用 set palette maxcolors <N>，将 N 设置小一点 避免这个问题。这个命令会从调色板中产生 N 个不同的色彩。如果你希望色彩不是等间距的那么你需要 set palette defined 命令。

RGB 色彩模式可能对某些工作并不合适。使用 model 可以设置色彩模式为：RGB HSV CMY YIQ XYZ。我们的讨论基于 RGB，但是实际上其他色彩模式也是适用的：R 可代指 H C Y X。

Rgbformulae

它需要 3 个逗号分割的整数，它们介于 -36 到 36 之间，这些整数代表了一个函数的编号。show palette rgbformulae 你可以看到：

```
gnuplot> show palette rgbfor
* there are 37 available rgb color mapping formulae:
  0: 0                      1: 0.5                      2: 1
  3: x                      4: x^2                    5: x^3
  6: x^4                    7: sqrt(x)                  8: sqrt(sqrt(x))
  9: sin(90x)               10: cos(90x)               11: |x-0.5|
 12: (2x-1)^2               13: sin(180x)              14: |cos(180x)|
 15: sin(360x)              16: cos(360x)              17: |sin(360x)|
 18: |cos(360x)|            19: |sin(720x)|            20: |cos(720x)|
 21: 3x                     22: 3x-1                   23: 3x-2
 24: |3x-1|                 25: |3x-2|                 26: (3x-1)/2
 27: (3x-2)/2               28: |(3x-1)/2|             29: |(3x-2)/2|
 30: x/0.32-0.78125         31: 2*x-0.84               32: 4x;1;-2x+1.84;x/0.08-11.5
 33: |2*x - 0.5|           34: 2*x                    35: 2*x - 0.5
 36: 2*x - 1
```

set palette rgbformulae 7,5,15 表示 $f(x) = \text{RGB}(\sqrt{x}, x^3, \sin(360x))$ 。x 是灰度值，对应点的 z 值在 minz 和 maxz 所占的比例。负数代表对应的反色。

比较不错的 RGB 色彩设置：

```
7, 5, 15      ... traditional pm3d (black-blue-red-yellow)
3, 11, 6      ... green-red-violet
```

```

23,28,3 ... ocean (green-blue-white); try also all other permutations
21,22,23 ... hot (black-red-yellow-white)
30,31,32 ... color printable on gray (black-blue-violet-yellow-white)
33,13,10 ... rainbow (blue-green-yellow-red)
34,35,36 ... AFM hot (black-red-yellow-white)
完整的 HSV 色彩空间
3,2,2 ... red-yellow-green-cyan-blue-magenta-red

```

rgbformulae 指定的映射会同样作用到 HSV,XYZ,等模式中，它并不只限于 RGB 模式。

Defined

灰度到 RGB 的映射可以使用 set palette defined 手动设置。

语法：

```
set palette defined { ( <gray1> <color1> {, <grayN> <colorN>}... ) }
```

<gray>是灰度值，它会被映射到 $[0,1]$ ¹。<color>是色彩，色彩可以用多种方式书写：<r> <g> ，色彩名字，#rrggbb；<r> <g> 必须是 $[0,1]$ 。色彩名和#rrggbb是字符串。色彩名根据 show colormnames 所显示的列表转换成色彩值。#rrggbb是16进制色彩。

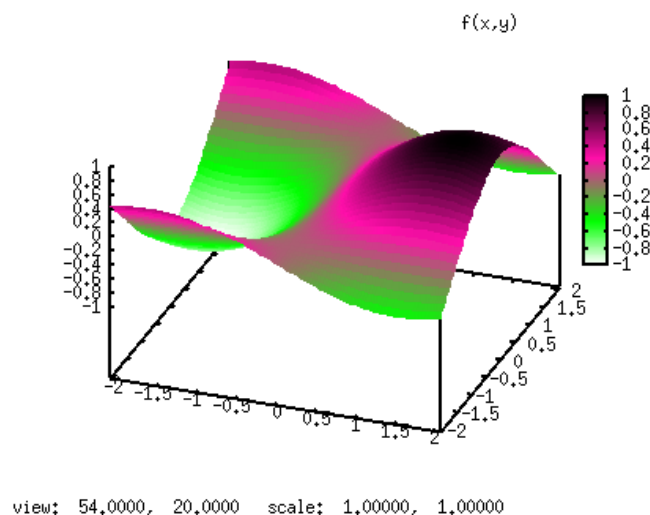
你可以在 palette defined 中混合使用色彩表示法。

例如：

```

set palette defined (0.1 'white' , 0.3 0 1 0,0.6 '#ff0faa',0.9 'black')
f(x,y)=sin(x)*cos(y)
splot f(x,y) with pm3d

```



1 映射方法：set palette defined (0 "blue", 3 "green", 6 "yellow", 10 "red")
blue 在灰度 0 green 在灰度 3/10 yellow 在灰度 6/10 red 在灰度 100%

如果 `set palette defined ()` 没有定义任何灰度,那么会使用默认的全光谱映射. 使用 `show palette gradient` 显示灰度和色彩映射表.

```
gnuplot> set palette defined (0.1 'white' , 0.3 0 1 0, 0.6 'ff0faa', 0.9 'black')
gnuplot> show palette gradient
0. gray=0.0000, (r,g,b)=(1.0000,1.0000,1.0000), #ffffff = 255 255 255
1. gray=0.2500, (r,g,b)=(0.0000,1.0000,0.0000), #00ff00 = 0 255 0
2. gray=0.6250, (r,g,b)=(1.0000,0.0588,0.6667), #ff0faa = 255 15 170
3. gray=1.0000, (r,g,b)=(0.0000,0.0000,0.0000), #000000 = 0 0 0
```

使用这张表你可以在其他图形软件中导入色彩设置。

实例：

定义调色板：

```
set palette model RGB
set palette defined ( 0 "black", 1 "white" )
```

色彩标尺，蓝到红到黄的色彩渐变，以下三条命令等价：

```
set palette defined ( 0 "blue", 1 "yellow", 2 "red" )
set palette defined ( 0 0 0 1, 1 1 1 0, 2 1 0 0 )
set palette defined ( 0 "#0000ff", 1 "#ffff00", 2 "#ff0000" )
```

让调色板使用彩虹色：

```
set palette defined ( 0 "blue", 3 "green", 6 "yellow", 10 "red" )
```

HSV 色彩空间，全色域：

```
set palette defined ( 0 0 1 1, 1 1 1 1 )
set palette model HSV
set palette defined ( 0 0 1 0, 1 0 1 1, 6 0.8333 1 1, 7 0.8333 0 1)
```

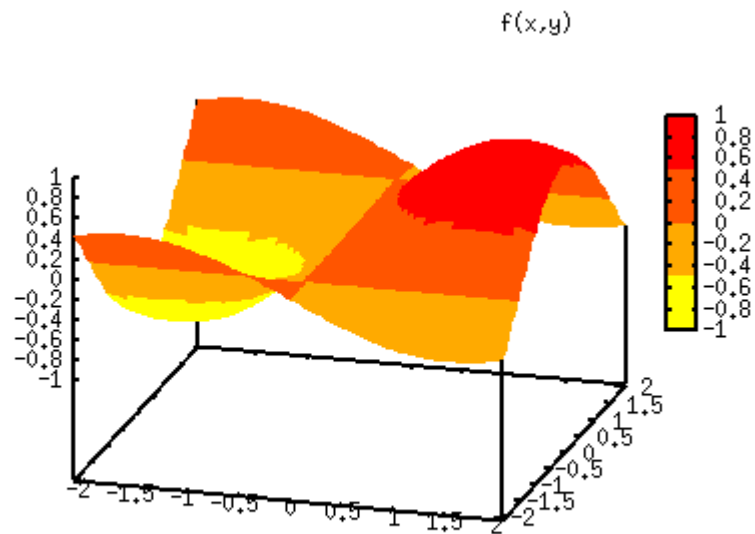
产生类似于 MATLAB 默认调色板：

```
set pal defined (1 ' #00008f' , 8 ' #0000ff' , 24 ' #00ffff' , \
                40 ' #ffff00' , 56 ' #ff0000' , 64 ' #800000' )
```

产生只有少数色彩的调色板：

```
set palette model RGB maxcolors 4
set palette defined ( 0 "yellow", 1 "red" )
```

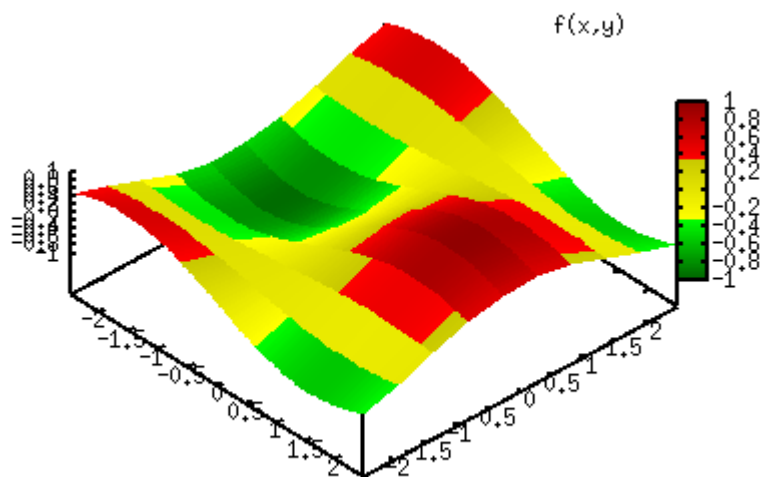
效果如下图，只有 4 个色彩：



view: 65.0000, 16.0000 scale: 1.00000, 1.00000

交通灯色彩 (不是平滑过度, 在灰度 1/3 2/3 有跳跃):

```
set palette model RGB
set palette defined (0 "dark-green", 1 "green", \
                    1 "yellow", 2 "dark-yellow", \
                    2 "red", 3 "dark-red")
```



view: 26.0000, 47.0000 scale: 1.00000, 1.00000

function

set palette functions <Rexpr>,<Gexpr>,<Bexpr> 选项设置灰度映射到彩色时的计算方法。输入值和返回值都必须在[0:1]。

实例：

全色域调色板：

```
set palette model HSV functions gray, 1, 1
```

黑色到金色:

```
set palette model XYZ functions gray**0.35, gray**0.5, gray**0.8
```

伽玛校正黑白调色板:

```
gamma = 2.2  
color(gray) = gray**(1./gamma)  
set palette model RGB functions color(gray), color(gray), color(gray)
```

cubehelix

如果你不知道什么是 cubehelix, 就不用看这个解释了。

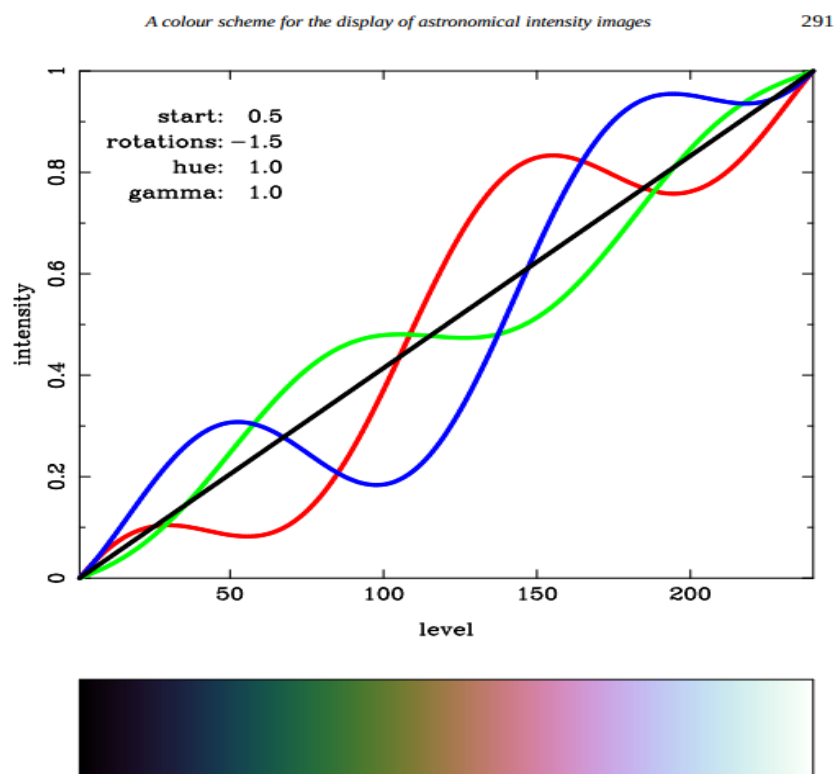
Cubehelix 选项定义一个调色板集合。根据灰度值, 色彩沿着标准色轮变化, 亮度也是单调变化,

D A Green (2011)

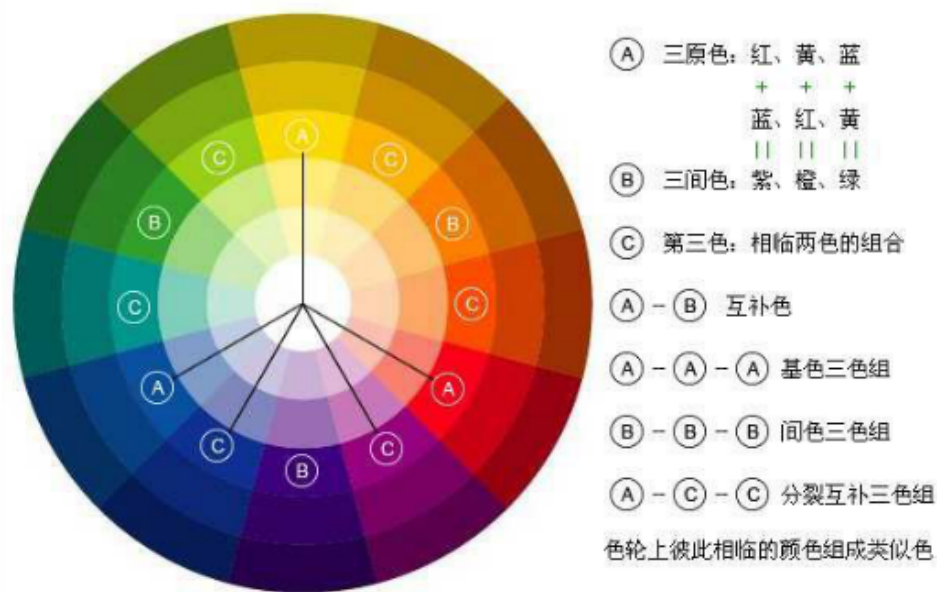
<http://arxiv.org/abs/1108.5083>

<http://arxiv.org/pdf/1108.5083v2.pdf>

cubehelix 参考资料, 由 start ,gamma rotations 不同会产生不同的调色板:



标准色轮:



色轮的色彩值由弧度决定。Start 用弧度定义了色彩开始点。Cycles 定义了多少色轮圆会被扫描。Saturation 的值决定色彩饱和度，值越大色彩越饱和。Saturation >1 导致剪切 RGB，并且亮度也非线性变化。调色板也受到 set palette gamma 影响，其默认值为：

```
set palette cubehelix start 0.5 cycles -1.5 saturation 1
set palette gamma 1.5
```

file

Set palette file 导致辉阶信息从文件中读取。数据文件需要 using 指定 3 列或者 4 列输入列。当输入列为 3 时行号就被缩放到灰度[0:1]区间。

<filename> 可以是 '-'，它意为着从命令后面读取。参考 特殊文件名 78。

实例：

从文件中读取到 RGB：

```
set palette file 'some-palette' using ($1/255):($2/255):($3/255)
```

等距彩虹调色板：

```
set palette model RGB file '-'
0 0 1
0 1 0
1 1 0
1 0 0
e
```

也可以使用二进制文件参考：二进制关键字：67

实例：输入 64 个 RGB double 三原色到调色板，

```
set palette file 'palette.bin' binary record=64 using 1:2:3
```

伽玛校正

对于灰度映射，伽玛校正可以使用 `set palette pamma <gamma>` 开启。`<gamma>` 默认为 1.5，这个值对于大多数输出终端效果都不错。

伽玛校正被用于 cubelix 调色板，不应用在其他调色板上。无论如何，你可以使用函数去控制伽玛映射。

实例：

```
set palette model RGB
set palette functions gray**0.64, gray**0.67, gray**0.70
```

如果希望在手动调色板上使用伽玛校正，应该在调色版合适位置设置合适的色彩。

使用命令：

```
set palette defined ( 0 0 0 0, 0.5 .73 .73 .73, 1 1 1 1 )
```

请不要使用这个：

```
set palette defined ( 0 0 0 0, 1 1 1 1 )
```

对于更多的插入色彩，伽玛校正自动拟合它们，使得校正更好。

Postscript

为了减少 ps 文件的尺寸，只有灰度值和部分 rgb 值被写进文件。解析公式用 postscript 语言编码到文件开始，紧临 pm3d 的绘制代码。通常，它意味着只写入 3 个被使用的公式是有意义的。但是对于 multiplot，或者其他原因，你需要手动编辑 ps 文件内的转换函数。默认选项 `nops_allcF`。写入所有公式到 ps 文件。你也许对为了使用多个 palettes 而手动编辑 ps 文件的技巧感兴趣。你可以使用 multiplot 和 origin size 关键字，产生类似效果。

如果 pm3d 映射根据网格或者规整的数据绘制，那么嵌入 pm3dCompress.awk 脚本可以减少 ps 文件体积，最多达 50%。

使用方法，

```
awk -f pm3dCompress.awk infile.ps > outfile.ps
```

如果 pm3d 映射根据矩形网格数据绘制，那么嵌入 pm3dConvertToImage.awk 脚本可以有效降低 ps 文件体积。

使用方法：

```
awk -f pm3dConvertToImage.awk <thefile.ps >smallerfile.ps
```

通过改变 `<maxcolors>` 可以限制输出色彩数目。

Pointintervalbox

线条的 `pointinterval` 属性可以用在 `linespoints` 绘制样式上。值为负数表示每 N 个画一次 point，并且每个点周边的小块区域(box)，用背景色填充。set `pointintervalbox` 控制填充 box 的半径。它的值用作因子，表示默认半径的倍数。例如，值为 2 表示，默认半径的 2 倍。

Pointsize

set `pointsize` 控制 points 相关的绘制样式，所绘制的点的大小。它的值用作因子，表示默认 size 的倍数。

语法

```
set pointsize <multiplier>
show pointsize
```

默认因子是 1。点拥有更大的尺寸使得点更明显。

使用：

```
plot f(x) with points pointsize 3
```

plot 命令使用 `pointsize` 关键字，只在某一个 plot 命令中修改 `pointsize`，而不影响其他 plot。

Polar

set `polar` 命令，把直角坐标系，改变成极坐标系。

语法：

```
set polar
unset polar
show polar
```

在极坐标系中，自变量的符号发生改变，不再是 x y 。而是 t 和 u ， t 表示角度：弧度 $[0:2\pi]$ 角度 $[0:360]$ 。角度单位由 `set angles` 控制，参见：93。

`unset polar` 将坐标系恢复成直角坐标系。

set `polar` 命令不支持 `splot`。`splot` 使用 `set mapping` 产生类似的功能。

在极坐标系中函数表达式应该类似与： $r=f(t)$ 。 t 表示角度， r 表示半径。`xrange` `yrange` 也不再有效，应该使用：`set trange $[-2\pi:2\pi]$` ，`set rrange $[0:3]$` 。

实例：

```
set polar
plot t*sin(t)
set trange  $[-2\pi:2\pi]$ 
set rrange  $[0:3]$ 
plot t*sin(t)
```

上面的例子，第一个 plot 命令使用默认定义域 $[0:2\pi]$ ，值域 rrange 会自动缩放。第二条 plot 命令值域和定义域都限制住了。

set size square 使得图像的轴拥有相同的缩放比例，使得圆形看起来是圆的。

参考实例：polar.dem poldat.dem

print

set print 命令重定向 print 命令的输出到文件。

语法：

```
set print
set print "-"
set print "<filename>"
set print "<filename>" append
set print "|<shell_command>"
```

没有参数时，输出被定向到标准错误流。 "-" 表示输出在标准输出。 append 表示以添加模式输出。 "|<shell_command>" 表示利用管道重定向到一个 shell 命令。

psdir

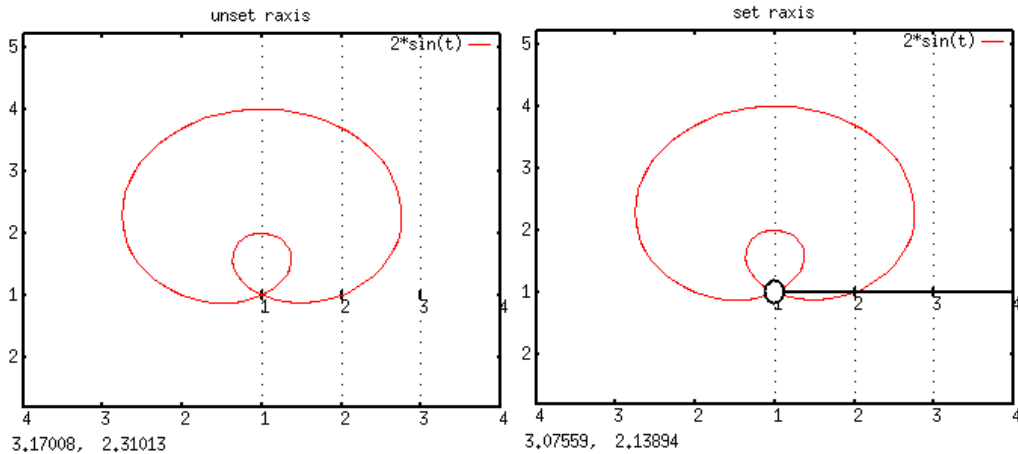
set psdir <directory> 命令 控制 postscript 终端搜索 prologue.ps 和字符编码文件所使用的路径。你能够使用这个选项使得 ps 文件使用不同的配置。搜索顺序是：

- 1) set psdir 命令指定的任何路径。
- 2) 环境变量 GNUPLOT_PS_DIR 指定的路径
- 3) 系统默认文件夹中的第一个或者某一个。
- 4) 'set loadpath' 指定的文件夹

raxis

当处在极坐标模式下，rrange 的范围不达到 0 时，你可能需要用某种方法来表示出来。set raxis 会绘制一条水平轴，同时当 rrange 的最小值不为 0 时。会在图形中心绘制一个小圆。

命令 set raxis 和 unset raxis 用来开启和关闭这个功能。



Rmargin

set rmargin 设置图像的右页边距。查看 set margin 129.

rrange

set rrange 设置极坐标系，径向坐标的取值范围。这个功能和同时设置 set xrange 、set yrange 相同。set rrange [rmin:rmax] 等同与 设置 xrange yrange 在 $[-(rmax-rmin):+(rmax-rmin)]$ 。如果在 rrange 设置后再更改 xrange 和 yrange 不会再产生变化。对径向坐标的自动缩放必然导致 rmin=0 。

Rtics

set rtics 命令在极坐标系统的径向轴上放置刻度。它们只在 polar 模式下才显示出来。刻度只显示在原点的右边。mirror 关键字使得在原点左边也绘制对称的刻度。其他关键字参考 set polar 96和 set xtics 。

Samples

set samples 控制函数的采样率，数据插值。

语法：

```
set samples <samples_1> {,<samples_2>}
show samples
```

默认采样值在 100. 更高的采样率会产生更精确的图形，但是会消耗时间。这个选项不影响数据文件的绘制，除非你对数据文件使用了插值和逼近相关算法的关键字。参考：plot smooth 75,set dgrid3d 106 。

在绘制 2d 图形时只涉及到<samples_1> 。当绘制 3d 图形并且没有使用 hidden 相关选项时，采样率决定了样条曲线的密度。 <sample_1>决定 iso_v 上的采样密度，<sample_2>决定 iso_u 上的

密度。参考：set isosamples 119。

size

语法：

```
set size {{no}square | ratio <r> | noratio} {<xscale>,<yscale>}  
show size
```

<xscale> <yscale>是绘制图形所使用的缩放因子，它影响 图形 标签 页边距 的绘制。

特别注意：

早期版本的 gnuplot，在某些终端类型下使用这个值控制输出图形的画布大小。4.6 版本中不再有这个不确定性，几乎所有的终端都使用相同的语法控制输出画布大小：

```
set term <terminal_type> size <xx>,<yy>
```

控制画布大小为 xx,yy，不同输出终端拥有不同的取值范围。请参考手册中的终端部分。默认时，绘制会占满整个画布。

set size <xx>,<yy> 为图形相对与画布的比例。值为 1 表示图形正好占满画布；小于 1，图形大小等于画布乘上因子。大于 1 导致只有部分图形能够显示出来。最好不要设置因子大于 1，因为这在某些输出终端中会产生问题。

ratio 关键字导致 gnuplot 尝试创建一个 y 轴长度/x 轴长度=r 的图形。这个图形会出现在<xscale> <yscale>所决定的区域内。r 可以为负数。如果 r=-1，那么 gnuplot 尝试设置 x 和 y 的单位长度相同。这等价于 set view equal xy。r=-2，y 的单位长度是 x 的两倍。依次类推。。。

实例：

让图形占满整个画布：

```
set size 1,1
```

绘制正方形，并且绘制区域为画布的一半，

```
set size square 0.5,0.5
```

设置图形高度为宽度的 2 倍：

```
set size ratio 2
```

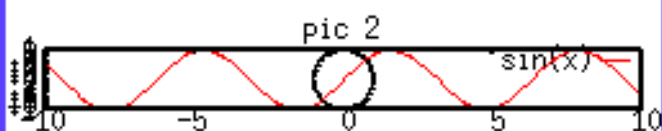
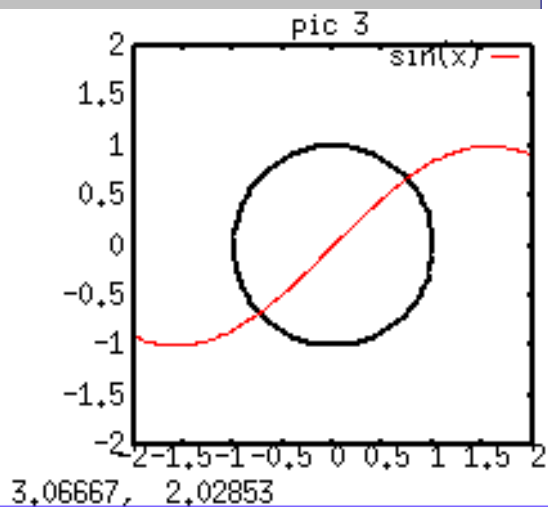
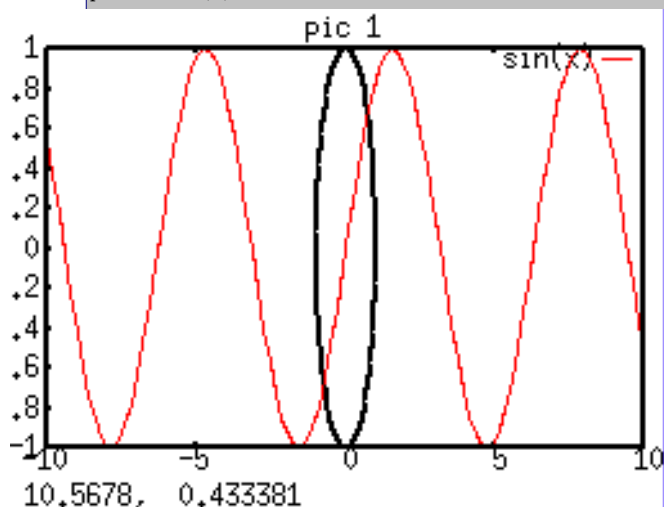
绘制理论上为圆形的椭圆，这段代码绘制了 3 副图像，观察它们的不同点：

```
reset  
pause -1 "pic1"  
set size 1,1  
set title "pic 1"  
set object ellipse at 0,0 size 2,2  
plot sin(x)  
pause -1 "pic 2"  
set size 1,1
```

```

set size ratio -1
set title "pic 2"
set object ellipse at 0,0 size 2,2
plot sin(x)
pause -1 "pic 3"
set size 1,1
set size ratio -1
set xrange [-2:2]
set yrange [-2:2]
set title "pic 3"
set object ellipse at 0,0 size 2,2
plot sin(x)

```



pic 1 : 你可能期望长轴和短轴半径相同的椭圆应该表现为圆形，但是图形中看起来并不是这样。长轴和短轴分别使用 x 和 y 的坐标值，因此当 x 和 y 的单位长度不同时相同的数值在两个轴上表现出不同的长度，所以看起来不是圆形。因此你需要设置 x y 的单位长度相同。

pic 2 :

以下指令设置 x y 的单位长度相同。

```
set size ratio -1
```

由于 `plot sin(x)` 使用了自动缩放和默认的 `xrange`、`yrange`，导致 `xrange [-10:10]` `yrange [-1:1]` 所以 `pic 2` 看起来很不爽。

pic 3：在 `pic2` 基础上加入 `set xrange [-2:2]` 和 `set yrange [-2:2]`，这样圆形就比较美观了。

这个例子没大用，帮助读者理解 `set size` 对图形的影响。

Style

`set style` 命令用于设置默认的绘制样式。`set style data`、`set style function` 分别设置对于数据文件的绘制样式和函数的绘制样式。参考：`plot with 87`。完整的样式列表参阅：`29`。

语法：

```
set style function <style>
set style data <style>
show style function
show style data
```

对于某个样式的默认属性也可以设置：

```
set style arrow <n> <arrowstyle>
set style fill <fillstyle>
set style histogram <histogram style options>
set style line <n> <linestyle>
```

如果你的 `gnuplot` 编译时支持了 `object`，那么以下命令可以发挥作用：

```
set style rectangle <object options> <linestyle> <fillstyle>
set style circle radius <size>
set style ellipse size <size> units {xy|xx|yy}
```

set style arrow

每个输出终端都预定义了一系列箭头和点样式，使用 `test` 命令可以查看她们。`set style arrow` 定义一个列表这让你在后来可以通过样式编号来指定样式。参考：`18`，`arrow`

语法：

```
set style arrow <index> default
set style arrow <index> {nohead | head | heads}
                        {size <length>,<angle>{,<backangle>}}
                        {filled | empty | nofilled}
                        {front | back}
                        { {linestyle | ls <line_style>}
                          | {linetype | lt <line_type>}
                          {linewidth | lw <line_width> }
unset style arrow
```



```
show style arrow
```

<index> 就是样式的 ID，你可以在 plot 命令中使用这个 ID，来引用它所代表的 arrow 属性。

default 关键字会设置指定 ID 的样式到默认值。

如果<index> 已经存在，只有给定属性值被改变，其他属性值保持不变。如果<index>不存在，没有给定值的属性被设置为默认值。

指定 nohead 产生一个箭头没有^尖帽——也就是一个线段。默认时箭头有一个头部，使用 heads 关键字使得头尾都有箭头。相关定义参考，set arrow 93。

size，filled empty front back 等关键字的使用见：set arrow 93。

实例：

定义一个 ID 为 1 的无箭头 宽度为 2 的箭头样式，并且在 set arrow 中使用这个样式。

```
set style arrow 1 nohead lw 2
set arrow arrowstyle 1
```

boxplot

set style boxplot 命令控制所有与 box(绘制样式包含的矩形元素)相关的属性，特别是 boxplot 样式。

语法：

```
set style boxplot {range <r> | fraction <f>}
                  {{no}outliers} {pointtype <p>}
                  {candlesticks | financebars}
                  {separation <x>}
                  {labels off | auto | x | x2}
                  {sorted | unsorted}
```

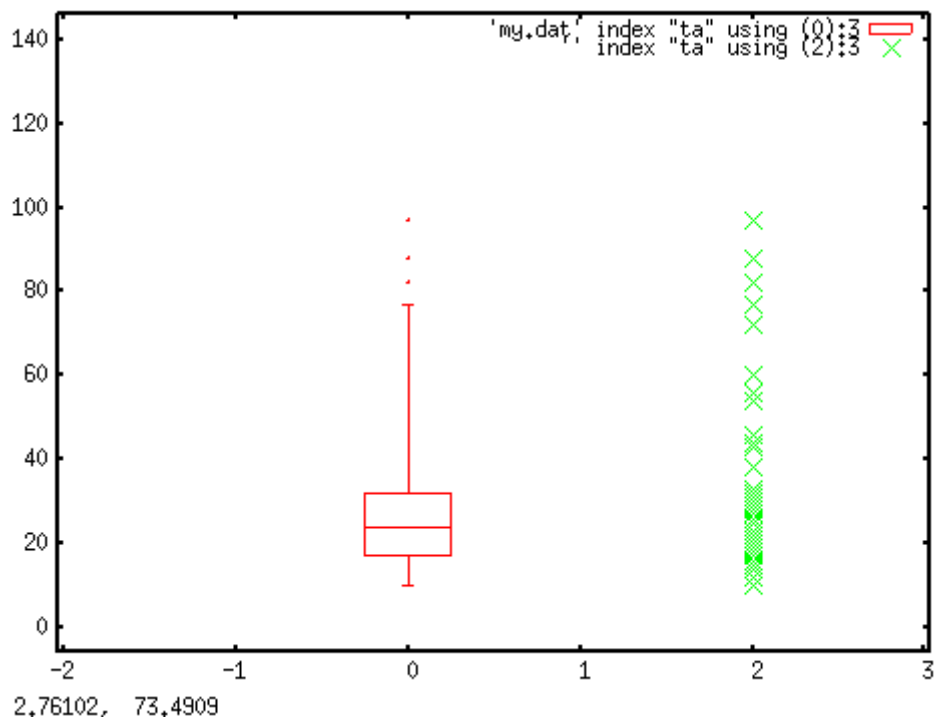
Boxplot 是一种表示统计分布的通用方法。四分线由如下规则确定，第一四分位数是 1/4 的点的值小于等于某个数，这个数叫做第一四分位数。2/4 的点的值小于等于某个值这个值称作第二四分位数。依次类推第三四分位数就是 3/4 的值小于等于它。矩形在第一四分位和第三四分位之间绘制。并且在第二四分位绘制一条水平线。矩形上方竖直绘制一个线段到用户限制值。参见：boxplot 绘制样式：31

默认时，boxplot 中的 box 上下两端有一个长为四分位距的 1.5 倍的须线。

boxplot 绘制统计图形，你可能希望它统计所有数据中的 95%主干数据，或者 98%的数据。

使用命令：

```
set style boxplot fraction 0.95
```



可以看到统计中囊括了 95% 的数据。剩下 5% 的部分绘制成单独的点。这些被 `fraction` 参数排除的数据被称作 outlier，使用 `nooutliers` 关闭 `fraction` 选项。

`candlesticks` 和 `finance` 用来控制图像是以蜡烛图显示还是金融图形。

当 `using` 使用了 4 个输入列那么第四输入列被当作数据标签，第一列和第三列分别作为起始横坐标和宽度处理，最好使用常数。gnuplot 会把所有拥有相同标签的数据放到一个 boxplot 中绘制，因此会产生多个 boxplot，标签会自动显示在 x 轴刻度上。boxplot 之间距离为 1，使用 `separation` 选项更改它。

`labels` 选项控制盒须图的标签如何设置。默认时 标签被放置到水平轴上，与 `plot` 命令的 `axis` 选项保持一致。它实际等于 `labels auto`。选项 `labels x` 和 `labels x2` 使得标签出现在 x 或 x2 上。`labels off` 关闭标签。默认时 boxplot 根据文件中标签出现的先后顺序绘制。使用 `sorted` 或者 `unsorted` 选项使得先根据标签排序再绘制 boxplot。

`separation labels sorted unsorted` 选项只有在 boxplot 样式接受了 4 个输入列时才有效（因为只有这种情况才需要绘制多个 boxplot，才会使用到标签，间距等等）。

参考：boxplot 31 candlesticks34 financebars 40

set style fill

`set style fill` 命令用来控制 box、histograms、candlesticks、filledcurves 的默认填充。这个选项的值会被明确使用填充选项的 `plot` 命令忽略。

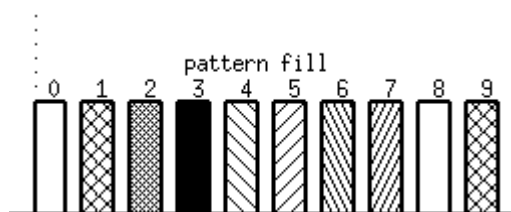
语法：

```
set style fill {empty
  | {transparent} solid {<density>}
  | {transparent} pattern {<n>}}
{border {lt} {lc <colourspec>} | noborder}
```

默认的填充是 empty。

solid 选项让填充色为实心色彩。<density>参数控制填充色的密度。<density>==0 时 box 不被填充。值为 1 时，填充色为当前绘制样式的线条色。如果没有<density> 默认为 1

pattern 选项。你可能希望使用网格填充、斜线填充、反斜线填充、点装填充等等。使用 pattern 选项使用内置图样进行填充。可用的图样类型和数目依赖与指定终端，使用 test 命令查看。如果多个数据集合需要使用填充 box，那么模式循环从<n>开始，依次遍历所有可用的图样，类似于在多次 plot 上线条样式的自动变化。



empty 选项，不进行填充。这是默认值。

border 选项，box 有实心边框，使用当前线条类型。border <colourspec> 可以设置一个指定的边框色彩。

set style fill transparent

某些终端允许使用透明属性。开启 transparent 选项时，density 参数被当作"可见性"。值为 0 完全透明，值为 1 完全可见。在使用开启 transparent 的图案填充时，模式的背景只能是完全透明或者完全可见。

终端	Solid	Pattern	pm3d
Gif	no	yes	no
jpeg	yes	no	yes
pdf	yes	yes	yes
png	truecolor	index	yes
post	no	yes	no
svg	yes	no	yes
win	yes	yes	yes
wxt	yes	yes	yes

x11	no	yes	no
-----	----	-----	----

注意，对于拥有透明填充的图像，在显示和设计的时候有一些附加的限制。例如，png 终端，只能在 truecolor 选项开启的时候才支持透明填充。某些 pdf 浏览器并不能正确显示包含透明的图像。Ghostscript/gv 不能正显示图案填充区域，当然严格的 PostScript 打印机是可以工作的。

Set style function

Set style function 命令可以改变绘制函数时的默认绘制样式。

语法：

```
set style function <plotting-style>
show style function
```

set style increment

注意：这个命令已经被丢弃，使用 set linetype 完成同样的功能。参考: set linetype 126

syntax:

```
set style increment {default|userstyles}
show style increment
```

By default, successive plots within the same graph will use successive linetypes from the default set for the current terminal type. However, choosing set style increment user allows you to step through the user-defined line styles rather than through the default linetypes.

Example:

```
set style line 1 lw 2 lc rgb "gold"
set style line 2 lw 2 lc rgb "purple"
set style line 4 lw 1 lc rgb "sea-green"
set style increment user
plot f1(x), f2(x), f3(x), f4(x)
```

should plot functions f1, f2, f4 in your 3 newly defined line styles. If a user-defined line style is not found then the corresponding default linetype is used instead. E.g. in the example above, f3(x) will be plotted using the default linetype 3.

set style line

每个终端都预定义了线条样式和点样式的集合，使用 test 命令可以查看她们。Set style line 定义线条类型和宽度和点样式、点大小。定义后的样式可以在后来的 plot 语句中指定样式 ID 来引用相关属性。

语法：

```
set style line <index> default
set style line <index> {{linetype | lt} <line_type> | <colourspec>}
    {{linecolor | lc} <colourspec>}
    {{linewidth | lw} <line_width>}
    {{pointtype | pt} <point_type>}
    {{pointsize | ps} <point_size>}
    {{pointinterval | pi} <interval>}
    {palette}
unset style line
show style line
```

default 设置所有线条样式到默认状态。

如果<index>已经存在，只改变指定的选项值，其他值保持不变。

创建新的线条样式请不要修改默认的线条样式，因为创建过程需要默认的风格。自定义的线条样式会被 reset 命令删除。重定义线条样式本身请参考 set linetype 126。

线和点的类型(lt pt lc)默认被设置为<index>的值。为 lt pt lc 指定一个值则使用指定值。

线条宽度(lw)和点的大小(ps)是因子，真实的大小是当前终端默认宽度乘上这个因子。如果使用了 Set pointsize 那么忽略由 set style line 设置的 pointsize。

Pointinterval 控制 linespoints 绘制样式绘制的点之间的间隔。默认是 0，意味着每个点都绘制。例如：set style line N pi 3，对于 ID 为 N 的线条样式设置 pi 为 3，其他参数保持不变。它将导致 linespoints 样式，每第 3 个点绘制一次。不是所有输出终端都支持 linewidth 和 pointsize，如果终端不支持，它们会被忽略。

终端非依赖的色彩可以使用 linecolor <colourspec>或者 linetype <colourspec> 指定。<colourspec>参考：颜色指定 19。Set palette146,show colornames, cbrange

在 splot 命令中，palette 关键字等同与"linetype palette z"

实例：默认的线条样式 1 2 3 分别代表红 绿 蓝，点的形状分别是 方 交叉 三角。

```
Set style line 1 lt 2 lw 2 pt 3 ps 0.5
```

定义一个新的线条样式，颜色为绿，宽度为默认的 2 倍，点的形状为三角形，点的大小为默认的一半。

```
set style function lines
plot f(x) lt 3, g(x) ls 1
```

绘制一个 f(x)的图像使用蓝色线条，g(x)图像使用用户定义宽度的绿色线条。类似的命令：

```
set style function linespoints
plot p(x) lt 1 pt 3,q(x) ls 1
```

绘制 q(x)的图像使用样式 1，它在前面定义成绿色 2 倍宽度。p(x)线条类型 1，它是红色。

```
splot sin(sqrt(x*x+y*y))/sqrt(x*x+y*y) w l pal
```

绘制一个曲面，使用 palette 调色板上色。注意它只在某些终端下有效。参考：set palette 146 ,set pm3d 141

```
set style line 10 linetype 1 linecolor rgb "cyan"
```

设置线条样式 10 为线条类型 1，颜色为 cyan。

Set style circle

语法：

```
set style circle {radius {graph|screen} <R>}
```

这个命令设置 circle 绘制样式的默认半径,它会被应用在函数的绘制中，或者应用在只有两个输入列时。默认值是：set style circle radius graph 0.02

set style rectangle

设置默认的矩形属性。Set object 命令默认使用它，但是 set object 也可以手动设置矩形相关属性。

语法：

```
set style rectangle {front|back} {lw|linewidth <lw>}  
    {fillcolor <colourspec>} {fs <fillstyle>}
```

参考：颜色指定 19 填充样式 162

实例：

```
set style rectangle back fc rgb "white" fs solid 1.0 border lt -1  
set style rectangle fc linestyle 3 fs pattern 2 noborder
```

默认值是使用背景色实心填充并且使用黑色边框。

set style ellipse

语法：

```
set style ellipse {units xx|xy|yy}  
    {size {graph|screen} <a>, {{graph|screen} <b>}}  
    {angle <angle>}
```

这个命令控制默认的椭圆轴使用的单位。默认是 xy，它意味着椭圆横轴使用 x 的单位长度。椭圆竖轴使用 y 的单位长度。在这种情况下椭圆轴实际的比值依赖于它的单位长度。当设置 units 为 xx 或者 yy 时长轴和短轴使用相同的单位长度，这意味着长轴和短轴的数值比等于它在图形上的长度比值。

此选项为全局设置，它影响 set object 绘制的椭圆也影响 plot 绘制的椭圆。当然 set object 和 plot 命令都可以直接使用 units 选项。

使用 size 关键字设置椭圆长轴和短轴默认长度。这个默认的长度用于只有两个输入列时绘制椭圆的 plot

命令。

```
set style ellipse size graph 0.05,0.03
```

设置默认长轴为图像高度的 0.05 短轴为 图像高度的 0.03。

angle <orientation> 设置旋转角度，角度值等于主轴和 x 轴的夹角。角度单位是 度，而不是弧度。

Surface

set surface 命令控制 splot 命令是否显示曲面。

语法：

```
set surface
unset surface
show surface
```

曲面在某些绘制样式下被绘制。unset surface 导致 splot 命令不根据数据或者函数绘制点或者线。如果你希望在某些个别的 splot 中不绘制曲面可以使用 nosurface 关键字。等高线依旧会被绘制在曲面上，它受 set contour 选项控制。

unset surface ; set contour base 这两条命令合用。对于网格数据显示等高线很有用。

Table

当 table 模式开启时，plot 和 splot 命令会在标准输出 显示 ASCII 表格，格式为：X Y {Z} R。字符 R 有 3 种值："i" 表示点在有效区域。"o" 表示点超出有效区域。"u" 点无定义。数据格式按照坐标轴的标签格式输出(参见：set format)。字段之间使用空白分割。

set table 可以输出所绘制的点。你可以把坐标保存到文件中，用其他程序处理。

语法：

```
set table {"outfile"}
plot <whatever>
unset table
```

列表输出到指定的"outfile"文件中。如果没有指定文件，就输出到 set output 指向的目标文件中。必须明确使用 unset table 才能关闭 table 模式。

Terminal

gnuplot 支持多种图形终端。set terminal 命令告诉 gnuplot 使用哪种类型的输出。

语法：

```
set terminal {<terminal-type> | push | pop}
show terminal
```

如果 <terminal-type> 没有指定，gnuplot 会输出可用的终端列表。<terminal-type> 可以被缩写。

如果 set terminal 和 set output 同时使用。你最好把 set terminal 放在 set output 前面。因为某些 OS 上终端会设置一些标记。

一些终端有一些附加的选项。如果先使用：set terminal <terminal-type> <options>，随后再次使用 set terminal <term> 不会重设已存在的选项值。

命令 set term push 保存当前终端的选项值。使用 set term pop 会读取，上次 push 的选项。它等于：save term 和 load term，但是不访问文件系统。

gnuplot 启动后，默认终端或者来自启动文件的终端设置会被自动 push。

Termoption

set termoption 命令允许你改变终端行为，但是不需要 set terminal。每条命令只能设置一个选项。并且只有一部分选项可以使用 set termoption 设置。

```
set termoption {no}enhanced
set termoption font "<fontname>{,<fontsize>}"
set termoption fontscale <scale>
set termoption {solid|dashed}
set termoption {linewidth <lw>} {lw <lw>}
```

tics

控制所有坐标轴的主刻度。

单独设置某个坐标轴的刻度使用 set xtics 或者 set x2tics 等等。unset tics 关闭坐标轴刻度。

语法：

```
set tics {axis | border} {{no}mirror}
    {in | out} {scale {default | <major> {,<minor>}}}
    {{no}rotate {by <ang>}} {offset <offset> | nooffset}
    {left | right | center | autojustify}
    { format "formatstring" } { font "name{,<size>}" }
    { textcolor <colorspec> }
set tics {front | back}
unset tics
show tics
```

此 set 命令可以只作用于单独的坐标轴例如：x,y,z,x2,y2 和 cb。关键字 front 和 back 用于 2d 绘制中刻度是否在图形顶层。

axis 和 border 告诉 gnuplot 产生的刻度包括刻度点和对应的标签。axis 使得刻度点和标签始终出现在 x=0 和 y=0 这两条线上（它们就是原点坐标轴）。如果坐标轴非常靠近边界，axis 关键字会移动刻度标签到边界上。border 关键字使得刻度点始终出现在边界线上。典型的区别如下：

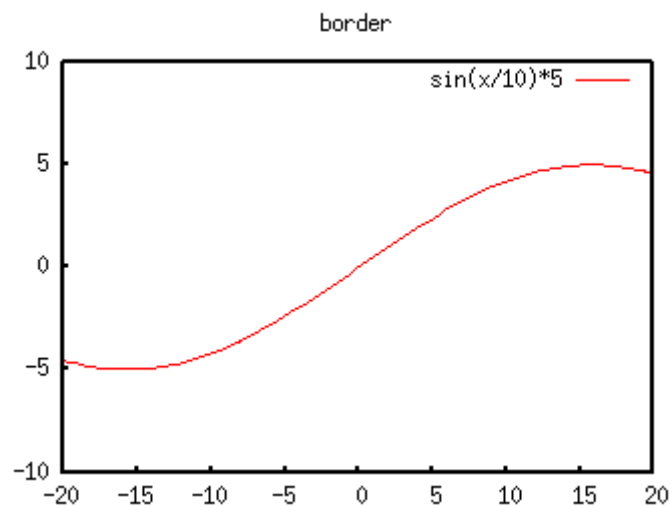
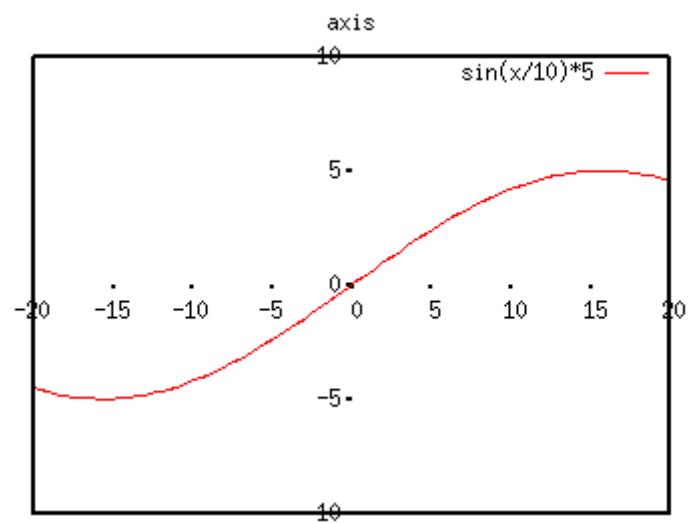
```
set xrange [-20:20]
set yrange [-10:10]
```



```

unset tics
set xtics axis
set ytics axis
set title "axis"
plot sin(x/10)*5
pause -1 "press any key"
set xtics border
set ytics border
set title "border"
replot

```



你可以搭配 set arrow 和 set tics axis 使图形美观。

```

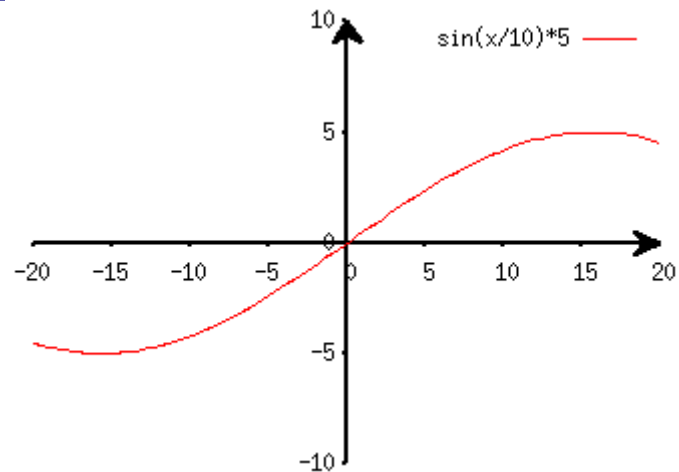
set xrange [-20:20]
set yrange [-10:10]
unset tics
set xtics axis
set ytics axis

```

```

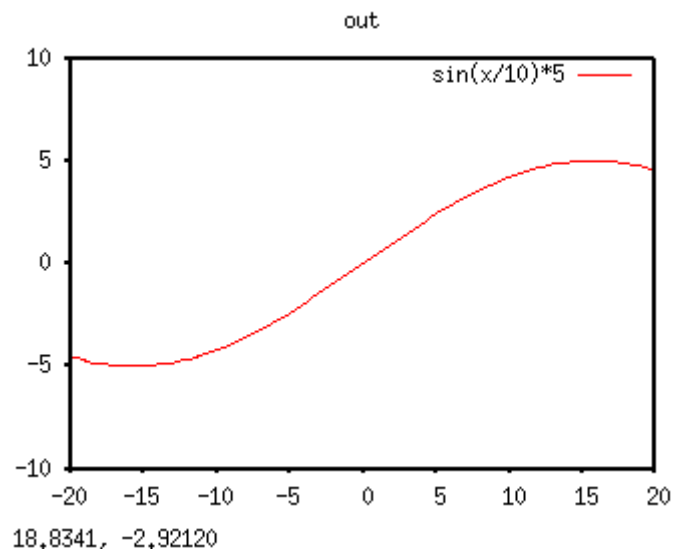
set title "axis"
unset border
set style arrow 1 head size 2,30,60 filled
set arrow from -20,0 to 20,0 as 1
set arrow from 0,-10 to 0,10 as 1
plot sin(x/10)*5

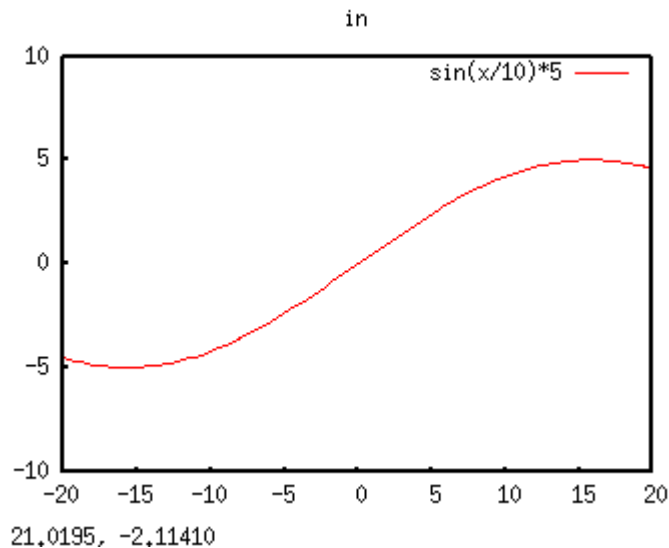
```



mirror 告诉 gnuplot 放置对称但没标签的的刻度到对面的坐标轴上：x1 和 x2 , y1 和 y2 相互对称。
 nomirror 不镜像刻度。

in 和 out 关键字放置刻度到边界内或边界外：





scale 关键字，控制刻度的大小。如果 <minor> 没有设置，那么它默认为 $0.5 * \text{<major>}$ 。默认的 <major> 为 1。

Rotate 告诉 gnuplot 旋转刻度标签 90 度。

默认情况下刻度标签根据坐标轴和旋转角度自动对齐，你可以使用 left right center，autojustify 控制对齐。

<offset> 控制标签出现的位置相对默认位置的偏移。格式为坐标 x,y 或者 x,y,z 并且可以使用 first second graph screen character 选择坐标系。

默认时刻度的标签自动对齐和旋转，使用 left right center autojustify 关键字可以控制它相对与刻度的对齐。其中 autojustify 就是默认的自动对齐。

set tics 无任何参数，恢复 tics 到默认设置。

Format 关键字控制刻度标签格式为 "formatstring"。

font 关键字设置字体："name{,<size>}"

textcolor 关键字设置文本色彩为：<colourspec>。

参考 set xtics 或者 set mxtics。

Ticslevel

已经丢弃，查看 set xyplane

ticscale

已经丢弃，使用 set tics scale。

timestamp

set timestamp 放置当前时间和日期到左页边距。

语法：

```
set timestamp {"<format>"} {top|bottom} {{no}rotate}
               {offset <xoff>{,<yoff>}} {font "<fontspec>"}
unset timestamp
show timestamp
```

<format> 是一个格式化字符串。它的默认值是 "%a %b %d %H:%M:%S %Y" ,星期，月份名，一个月的第几天，时 分 秒，四位数的年。可用的转义序列参见：gprintf 时间日期格式化占位符 112。使用 top bottom 关键字让文本出现在左页边距的上或者下。rotate 允许你将日期文本旋转 90 度，竖直排列。

<xoff> <yoff>分别设置相对默认位置的偏移。 是用来显示日期文本的字体。

set timestamp 可以缩写成 set time .

实例：

```
set timestamp "%Y-%m-%d %H:%M:%S" offset 80,-2 font "helvetica"
```

timefmt

此选项控制数据文件的日期数据的出现格式。只有当 set xdata time 开启时，或者使用 timecolumn (X)函数 此选项的值才被使用。目前版本的 gnuplot 控制数据中的时间格式，只有一个 timefmt 选项。这意味着如果你的数据文件中如果有多个时间字段，那么它们的格式应该相同。或者把日期作为字符串字段，然后使用.strptime (" format" ,stringcolumn(X)) 按照 format 格式解析第 X 字段的日期。

语法：

```
set timefmt "<format string>"
show timefmt
```

时间日期格式	
占位符	含义
%b	3 个字母的月份单词缩写

%B	月份单词全写
%d	月份数字 01-31
%H	小时 00-23 ，始终两个数字
%j	一年当中某天，1-366
%m	月份 01-12
%M	分钟 0-60
%S	秒数，0-60，输出是整数，读入是小数
%s	从 1970-01-01 00:00 UTC 到现在的秒数
%y	公元年低 2 位，0-99，有效年份：1969-2068
%Y	完整 4 位数字公元年。

任何字符都可以出现在格式字符串中，但必须严格匹配。`\t` 制表符 被识别。8 进制转义序列也被支持：`\127` 它表示 ASC 为 0127 的字符。如果在各个占位符之间没有分割字符`%d %m %y %H %M %S` 每个读取 2 个字符。如果`%S` 所占的数字后面紧接着小数点，那么它随后的小数也被读取。`%Y` 读取 4 个数字字符，`%j` 读取 3 个数字字符。`%b` 读取 3 个字符。`%B` 月份名称必须严格出现。

单个空格在格式字符串中代表 0 或更多的空白。`"%H %M"` 可以读取 1220 也可以读取 "12 20" 或者 "12 20" 。

对于日期格式在 `using n:n` 中可能分割成很多字段。例如，11:11 25/12/76 21.0 被分割成为 3 个字段。为了避免混乱，在数据中包含日期时，gnuplot 要求你必须使用 `using` 关键字。

使用 `timefmt` 读取 x 值时会根据模式跨越字段分隔符，例如数据如下：

```
01/06/93      1259    0.20    0.175
```

命令：

```
set timefmt "%d/%m/%y\t%H%M"
plot 'dat' using 1:3 w points, '' using 1:2:(sprintf("%d", $1)) with labels
```

观察上面的输出，使用`$1`得到的只有数字 1，但是在第一输入列使用字段 1，就会读取完整的时间。

如果日期数据包含单词表示的星期或者月份。格式化字符串必须避免这个。输出时间文本参考：`set format 110`。

当使用`%y`,从 2 个数字中读取年时 69-99 被当作 20 世纪，00-68 作为 21 世纪。这是根据:unix 98 的习惯。2 位数字表示年有千年虫问题（跨世纪日期运算），但是对于大部分数据是合适的。

参考：`set xdata` 和 日期和时间 28。

实例：

```
set timefmt "%d/%m/%Y\t%H:%M"
```

读取"30/12/2012 12:34" 这种格式的数据。但是注意，tab 字符必须严格匹配。

参考实例：

time data demo.

Title

set title 设置本次绘制的标题，它出现在整个画布的上方，居中对齐。set title 是一个特殊的 set label.

语法：

```
set title {"<title-text>"} {offset <offset>} {font "<font>{,<size>}" }  
      {{textcolor | tc} <colourspec> | default}} {{no}enhanced}  
show title
```

<offset> 必须以 x,y 或 x,y,z 的格式出现。坐标可以使用 first, second, graph, screen, or character 选择坐标系。默认使用 character 坐标系.例如："set title offset 0,-1 " 会把标签向下移动一个字符的高度。具体移动多少像素，取决于字体和输出终端。

 指定需要的字体。字体尺寸：<size> 取决于输出终端。

textcolor <colourspec> 设置文本色彩。参考：色彩设置 19. 和 调色板 146.

noenhanced 和 enhanced 关闭或者开启文本增强模式 8 .

set title 不接任何参数，清除当前的标题。

Tmargin

设置上方页边距。参见：set margin 129

trange

set trange 命令设置用于计算函数 $f(x,y)$ 的定义域范围。参考 set xrange

urange

set urange 和 set vrange 设置极坐标模式下自变量 $f(u,v)$ 定义域范围。参考 set xrange

variables

show variables 命令列出当前用户定义和系统内置的所有变量。Gnuplot 内置的变量都有前缀：GPVAL_ MOUSE_ FIT_ TERM_

语法：

```
show variables      # 显示没有前缀 GPVAL_ 的变量
show variables all  # 显示所有变量
show variables NAME # 显示变量 NAME 的值。
```

Version

show version 命令 显示当前 gnuplot 的版本和其他信息。

语法：

```
show version {long}
```

如果有 long 关键字，那么会显示 操作系统和编译选项。帮助文件的位置，和相关的 email 地址。

View

set view 命令设置 splot 命令的观察角度。它控制 3d 坐标系如何在平面窗口或者图片上显示。它可以控制旋转角度和缩放。用这个命令可以把 3d 图像投射成 2d 图片。

语法：

```
set view <rot_x>{, {<rot_z>} {, {<scale>} {, <scale_z>}}}
set view map
set view {no}equal {xy|xyz}
show view
```

<rot_x> <rot_z> 控制相对与初始角度的，旋转角度，单位是度。初始 0,0 角度，x 轴 落在图片下边界，y 轴落在图片左边界，z 轴完全垂直与可视平面。

命令 set view map 用来绘制投影。它常用于 pm3d 等高线的投影。

<rot_x> 限制在 [0:180] 之间。默认是 60 度。<rot_z> 限制在 [0:360] 默认是 30。<scale> 控制整个图像的缩放因子。<scale-z> 控制 z 轴的缩放因子。二者默认都为 1。

实例：

```
set view 60, 30, 1, 1
set view , , 0.5
```

你可以利用交互终端用鼠标旋转图形到合适角度，然后 show view 查看当前的 view 值,最后将它写进 gnuplot 脚本里面。

equal_axes

set view equal xy 强制 x 和 y 轴的 单位长度保持相同，并且缩放整个图形到画布内。set view equal xyz 缩放 z 轴，适应 x y 轴。这并不保证 zrange 在图像边界内。默认情况下，三个坐标轴各自缩放，使得整个坐标系出现在画布上。

Vrange

set urange 和 set range 命令 设置极坐标模式下的定义域范围，类似于直角坐标系下的 xrange yrange zrange。参考 set xrange

x2data

set x2data 命令设置 x2 轴数据是时间或者日期。参考：set xdata

x2dtics

set x2dtics 设置 x2 刻度为星期。参考 set xdtics

x2label

设置 x2 轴的标签，参考 set xlabel

x2mtics

设置 x2 刻度为月名。

x2range

设置 x2 轴的取值范围。参考：set xrange

x2tics

设置 x2 轴的主刻度，参考：set xtics

x2zeroaxis

绘制在 $y_2=0$ 处绘制虚线。参考 set zeroaxis

xdata

这个命令设置 x 轴的数据类型为时间或者日期。类似的 set ydata , zdata x2data y2data cbdata 也是设置对应轴的数据类型到时间或日期。

语法：

```
set xdata {time}  
show xdata
```

使用 time 参数，告诉设置类型数据类型为时间或者日期。没有 time 关键字，恢复数据类型为正常。结合 set timefmt 选项告诉 gnuplot 如何读取时间信息。时间数据在 gnuplot 中被转换成从本世纪开始到现在的秒数。目前版本的 gnuplot 控制数据中的时间格式，只有一个 timefmt 选项。这意味着如果你的数据文件中如果有多个时间字段，那么它们的格式应该相同。或者把日期作为字符串字段，然后使

用 `strftime("format",stringcolumn(X))` 得到第 X 字段所对应的日期值。此时 `xrange` 应该使用，字符串指定，否则可能被解释为 `gnuplot` 表达式：

```
set xrange ["2012-01-01 00:00":"2012-06-29 00:00"]。
```

`strftime()` 函数用来在刻度上产生标签。你可以把整个时间数据用引号括住成为字符串类型。也可以在坐标输入列直接使用 `timecolumn(X)` 得到第 X 字段的日期值，此时第 X 字段不应该是字符串。

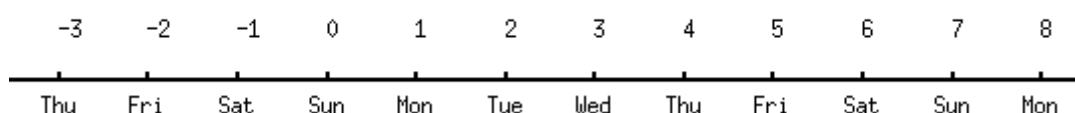
Xdtics

`set xdtics` 转化 x 轴的刻度到星期，0=sun ... 6=sat .超出[0:6]的整数值取 6 的模。`set noxdtics` 关闭坐标轴星期。

语法：

```
set xdtics
unset xdtics
show xdtics
```

类似功能的选项有：`set ydtics` `zdtics` `x2dtics` `y2dtics` `cbdtics`



xlabel

`set xlabel` 用于设置 x 轴的标题。类似的 `ylabel` `x2label` 等等 可用于其他坐标轴。

```
set xlabel {"<label>"} {offset <offset>} {font "<font>{,<size>}" }
      {textcolor <colourspec>} {{no}enhanced}
      {rotate by <degrees> | rotate parallel | norotate}
show xlabel
```

`<offset>` 用来设置相对默认标签位置的偏移。偏移使用 `x,y` 或者 `x,y,z` 格式指定，可以使用 `first`, `second`, `graph`, `screen`, 和 `character` 选择坐标系，参考：坐标 6。

```
set xlabel offset -1,0
```

会从默认位置向左移动一个字符宽度，具体的像素值跟字体有关。

`` 用来选择字体，`<size>` 用来选择字号。

`noenhanced` 关闭文本增强功能。参见：增强文本模式 8。删除标签使用 `set xlabel`。

```
set title "This Title"
set x2label "This is \nX2label"
```

`title` 的位置和 `x2label` 的位置差不多。

默认的标题都是水平放置，使用 rotate by <degrees> 。旋转指定角度。也使用 rotate parallel 旋转 90 度。

Set x2label 是内置的在轴旁边提供 label 的命令，你也可以使用 set margin 和 set label 手动添加任意标签。

Xmtics

Set xmtics 命令设置 x 轴的刻度为月份，1=Jan ... 12=Dec 。超出的值取 12 的模，

unset xmtics 关闭此选项。

语法：

```
set xmtics
unset xmtics
show xmtics
```

xrange

Set xrange 命令设置 x 轴的取值范围。类似的命令还有 urange vrange yrange zrange cbrange rrange trange x2range y2range。

语法：

```
set xrange { [{{<min>}:{<max>}}] {{no}reverse} {{no}writeback} }
           | restore
show xrange
```

reverse 逆转 x 轴的方向。Set xrange [0:1] reverse 设置 x 轴 左边值为 1 右边为 0 。

如果 range 中 <min> 或者 <max> 的值是 星号* 。那么对<min>或者 <max>开启自动缩放。语法是：

```
{XX < } * {< YY}
```

其中花括号的部分可以省略。它的含义是：开启自动缩放，但是缩放后的结果在 XX YY 范围之内。例如：

set xrange [0:12<*<200]

自动缩放 x 的上限，但是上限不超出：[12:200]

writeback 关键字保存自动缩放的范围到缓冲区。可随后读取。它用于你希望绘制多个曲线，但是只根据一个曲线缩放 xrange。writeback 关键字只在 plot 命令执行时被执行。因此必须出现在 plot 前方。读取 xrange 使用 restore。

实例：

```
set xrange [-10:10]
set yrange [*,*] writeback
```

```
plot sin(x)
set yrange restore
replot x/2
```

自动缩放 $\sin(x)$ 的 yrange , 然后固定这个 range , 绘制 $x/2$ 的图像。

设置 xrange 到默认区间[-10:10]:

```
set xrange [-10:10]
```

设置 yrange 递减:

```
set yrange [10:-10]
```

设置 z 的上限为 10 , 不影响下限 (下限可能被缩放) 。

```
set zrange [:10]
```

自动缩放 x 下线 , 不影响上限:

set xrange [*:]

自动缩放 x 下限但保持为正数 (可用于对数坐标) :

set xrange [0<*:]

自动缩放上下限制:

set xrange [*<10:50<*]

自动缩放上下限 , 持在 [-1000:1000]

set xrange [-1000<*:1000]

保证下线在-200 到 100 之间:

set xrange [-200<*<100:]

xtics

控制 x1 轴主刻度。使用 unset xtics 关闭刻度。 set xtics 开启刻度。类似的命令也可以用在 : ytics ztics,x2tics,y2tics cbtics,分别用于 y z x2 y2 cb 坐标轴。

语法 :

```
set xtics {axis | border} {{no}mirror}
    {in | out} {scale {default | <major> {,<minor>}}}
    {{no}rotate {by <ang>}} {offset <offset> | nooffset}
    {left | right | center | autojustify}
    {add}
    { autofreq
    | <incr>
    | <start>, <incr> {,<end>}
    | ({"<label>" <pos> {<level>} {,{"<label>"...} }
    { format "formatstring" } { font "name{,<size>}" }
```

```
{ rangelimited }  
  { textcolor <colorespec> }  
unset xtics  
show xtics
```

参见 set tics 对选项的解释包含插图，和实例代码。

axis 和 border 告诉 gnuplot 产生的刻度包括刻度点和对应的标签。axis 使得刻度点和标签始终出现在 $x=0$ 和 $y=0$ 这两条线上（它们就是原点坐标轴）。如果坐标轴非常靠近边界，axis 关键字会移动刻度标签到边界上。Border 关键字使得刻度点始终出现在边界线上。

对于 z 轴 axis 和 border 选项无效。

mirror 告诉 gnuplot 放置对称但没标签的的刻度到对面的坐标轴上： x_1 和 x_2 ， y_1 和 y_2 相互对称。nomirror 不镜像刻度。

in 和 out 关键字放置刻度到边界内或边界外。

scale 关键字，控制刻度的大小。如果 <minor> 没有设置，那么它默认为 $0.5 * \text{<major>}$ 。默认的 <major> 为 1。

Rotate 告诉 gnuplot 旋转刻度标签 90 度。rotate by <degree> 旋转指定角度。

默认情况下刻度标签根据坐标轴和旋转角度自动对齐，你可以使用 left right center，autojustify 控制对齐。

<offset> 控制标签出现的位置相对默认位置的偏移。格式为坐标 x,y 或者 x,y,z 并且可以使用 first second graph screen character 选择坐标系。

默认时刻度的标签自动对齐和旋转，使用 left right center autojustify 关键字可以控制它相对与刻度的对齐。其中 autojustify 就是默认的自动对齐。

set tics 无任何参数，恢复 tics 到默认设置。

Format 关键字控制刻度标签格式为 "formatstring"。

font 关键字设置字体："name{,<size>}"

textcolor 关键字设置文本色彩为：<colorespec>。

实例：

x 轴刻度向上移动一点：

```
set xtics offset 0,graph 0.05
```

刻度出现的位置，默认由 autofreq 选项自动产生。你可以通过两种方法手动产生：

1) <start>,<incr>,<end> 从 <start> 值开始间隔 <incr> 绘制刻度，直到 <end>。<incr> 可以为负数。如果在对数座标下 <incr> 被解释成等比因子。省略 <start> 会被当作负无穷，省略 <end> 会被当作正无穷

如果你需要指定负数，应该使用 0-XX 而不是 -XX。因为 gnuplot 的语法格式会造成语法错误：

```
set xtics border offset 0,0.5 -5,1,5
```

无法正确执行，因为 0.5 -5 被作为一个算数表达式计算，导致上面的表达式成为 offset 0,-4.5,1,5。会在最后一个逗号处提示表达式错误。

正确的做法：

```
set xtics border offset 0,0.5 0-5,1,5
```

或者

```
set xtics border offset 0,0.5 border -5,1,5
```

front back layerdefault 顾名思义，控制刻度显示在顶层或者底层。

实例：

在 0, 0.5, 1, 1.5, ..., 9.5, 10. 设置刻度

```
set xtics 0,.5,10
```

设置刻度 负无穷, -10, -5, 0, 5, 10, 正无穷

```
set xtics 5
```

在 1, 100, 1e4, 1e6, 1e8.

```
set logscale x; set xtics 1,100,1e8
```

在 0, 0.5, 1, 1.5, ..., 9.5, 10. 设置刻度

```
set xtics 0,.5,10
```

在 ..., -10, -5, 0, 5, 10, ... 设置刻度

```
set xtics 5
```

在 1, 100, 1e4, 1e6, 1e8. 设置刻度

```
set logscale x; set xtics 1,100,1e8
```

设置刻度在 1 10² 10⁴ 10⁶ 10⁸

```
set logscale x  
set xtics 1,100,1e8
```

2)

使用("<label>" <pos> <level>)格式可以，在值为<pos> 的位置产生刻度并且标签为 <label>。刻度不需要按照数值顺序指定。"<label>" 实际是格式化字符串，你可以不包含占位符。参考 gprintf 函数 111。如果没有<label> 那么使用 set format 的格式化字符串。

<level> 控制刻度等级，值为 0 表示主刻度，为 1 表示副刻度。

实例：

```
set xtics ("low" 0, "medium" 50, "high" 100)
```

```
set xtics (1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024)
set ytics ("bottom" 0, "" 10, "top" 20)
set ytics ("bottom" 0, "" 10 1, "top" 20)
```

在第二实例中，所有的刻度都有标签。在第三实例中，第二个刻度没有标签，在第四个实例中，没有被标记的刻度为副刻度。

实例：

```
set xtics 0, 5, 10
set xtics add ("Pi" 4.14159 )
```

有时候你希望在自动产生的刻度上添加一些刻度，你可以使用 `add` 关键字 再加上刻度。

无标签的副刻度可以使用 `set mtics` 绘制。

```
set mtics (1, 2, 3)
set mtics add (1, 2, 3)
```

xtics 时间数据

如果你希望自动产生的刻度是时间数据，时间必须使用字符串指定。字符串会使用 `timefmt` 解析成秒。并且坐标轴需要开启时间属性。

实例：

```
set xdata time
set timefmt "%d/%m"
set xtics format "%b %d"
set xrange ["01/12":"06/12"]
set xtics "01/12", 172800, "05/12"

set xdata time
set timefmt "%d/%m"
set xtics format "%b %d"
set xrange ["01/12":"06/12"]
set xtics ("01/12", "" "03/12", "05/12")
```

以上两段代码都会产生 "Dec 1" "Dec 3" "Dec 5" 但是在第二个代码中，"Dec 2"没有标签。

rangelimited

这个选项设置了刻度和图像边界的距离。

实例：

```
reset
set tics nomirror
set border 3
set yrange [0:10]
plot sin(x)+3
```

```
pause -1 "any"
set ytics nomirror rangelimited
refresh
```

图 1 :

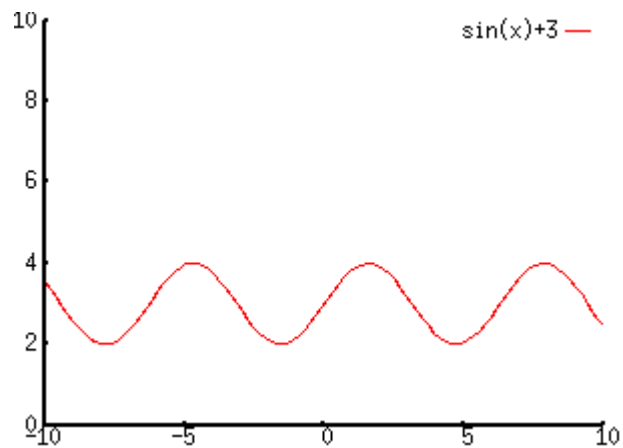
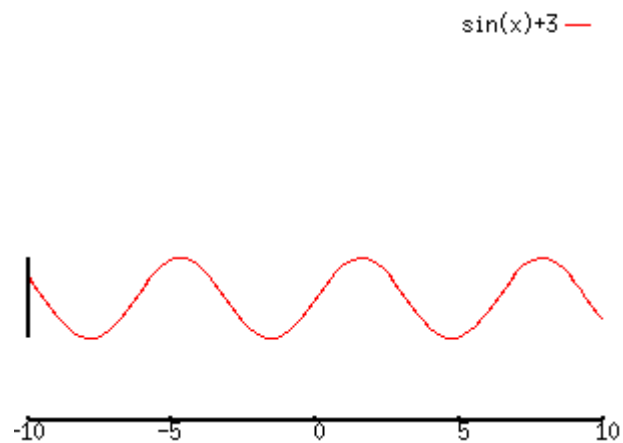


图 2 :



比较上图，区别很明显。

Xyplane

set xyplane 在 3d 绘制中设置 xy 平面出现的位置。

语法：

```
set xyplane at <zvalue>
set xyplane relative <frac>
set ticslevel <frac> # 等价于设置 xyplane relative
show xyplane
```

set xyplane at <z> 设置 xy 平面出现在指定 z 值。

set xyplane relative <frac> 设置 xy 平面出现 $z_{min} - (z_{max} - z_{min}) * \text{frac}$ 。

set ticslevel 已经抛弃。

实例：

```
reset
set view 68, 106
set view equal xyz
set xrange [-1.5:1.5]
set yrange [-1.5:1.5]
set ztics -1,0.4,1
print "this is:set xyplane at -1"
set xyplane at -1
splot sin(x*y)
pause -1 "next:set xyplane at 0.5"
set xyplane at 0.5
refresh
pause -1 "next :set xyplane relative 0.5"
set xyplane relative 0.5
refresh
```

xzeroaxis

在 $y=0$ 绘制虚线。参考 set zeroaxis

y2data

设置 y_2 轴的时间属性。参考：set xdata 176

y2dtics

设置 y_2 轴的刻度为星期，参考 set xdtics 176

y2label

设置 y_2 轴标签。参考 set xlabel 177

y2mtics

设置 y_2 刻度标签为月。参考 set xmtics 178

y2range

设置 y_2 取值范围，参考 set xrange 178

y2tics

设置 y_2 刻度。参考：set xtics 179

y2zeroaxis

在 $x_2=0$ 绘制虚线，参考 set zeroaxis

ydata

设置 y 轴的时间属性。参考：set xdata 176

ydtics

设置 y 轴的刻度为星期，参考 set xdtics 176

ylabel

设置 y 轴标签。参考 set xlabel 177

ymtics

设置 y 刻度标签为月。参考 set xmtics 178

yrange

设置 y 取值范围，参考 set xrange 178

ytics

设置 y 刻度。参考：set xtics 179

zeroaxis

在 $x=0$ 绘制虚线，参考 set zeroaxis

zdata

设置 z 轴的时间属性。参考：set xdata 176

zdtics

设置 z 轴的刻度为星期，参考 set xdtics 176

zlabel

设置 z 轴标签。参考 set xlabel 177

zmtics

设置 z 刻度标签为月。参考 set xmtics 178

zrange

设置 z 取值范围，参考 set xrange 178

zticks

设置 z 刻度。参考：set xtics 179

zeroaxis

在 $y=0, x=0$ 绘制平面。

Zero

设置 0 值临界点。

语法：

```
set zero <expression>
show zero
```

gnuplot 不会绘制在某个点"领域"以内的点。gnuplot 使用 zero 临界值判断领域。使用如果两个点距离在 zero 临界值以内，它们被认为是在计算误差以内，因此是相等的。

默认的 zero 值为 $1e-8$ 。最好不要设置大于 $1e-3$ 。当然设置为 0 也是不理智的。

Zeraxis

x 轴可以使用 set xzeroaxis 在 $y=0$ 处绘制。unset xzeroaxis 取消 $y=0$ 的绘制。

语法：

```
set {x|x2|y|y2|z}zeroaxis { {linestyle | ls <line_style>}
                           | { linetype | lt <line_type>}
                           { linewidth | lw <line_width>}}
unset {x|x2|y|y2|z}zeroaxis
show {x|y|z}zeroaxis
```

默认这个选项是关闭的。可以设置线条样式 线条类型 和线条宽度。如果没有指定线条样式默认使用 0。

实例：

绘制 $y=0$ 。

```
set xzeroaxis
```

绘制样式为 3 宽度为 2.5 的轴 $y=0$ 。

```
set xzeroaxis linetype 3 linewidth 2.5
```

shell

shell 命令 开启一个交互式 shell , 退出 shell 后自动返回到 gnuplot。

有两个办法执行单个 shell 命令。

1) ! Dir

2) system "dir"

```
system "date"; set time; plot "a.dat"
print=1; if (print) replot; set out; system "lpr x.ps"
```

splot

splot 进行 3d 绘制。splot 只有 x y z 三个轴 , 不同于 plot 命令 , 后者有 x2 y2 轴 , 前者没有。

语法 :

```
splot {<ranges>}
      {<iteration>}
      <function> | "<datafile>" {datafile-modifiers}}
      {<title-spec>} {with <style>}
      {, {definitions{,}} <function> ...}
```

默认时 xy 平面被绘制在 $z=zmin-0.5*(zmax-zmin)$ 。使用 set xyplane 修改它。大部分 splot 的参数和 plot 命令相同。当处在非 parametric 模式下 , <range> 按照 xrange yrange zrange 解析。处在 parametric 模式下 , 按照 urange vrange xrange yrange zrange 解析。

title 关键字 with 关键字 同 plot 一样。绘制样式只有 lines points linespoints dots impulses pm3d。

数据文件

splot 又有类似 plot 的数据文件关键字。

```
splot ' <file_name>' {binary <binary list>}
      {{nonuniform} matrix}
      {index <index list>}
      {every <every list>}
      {using <using list>}
```

splot 支持和 plot 相同的特殊文件名 : '-' 脚本内嵌入数据 。 '+' 自动采样 参考 特殊文件名 78。

binary matrix index every 关键字类似 plot 命令 参考 : plot 关键字 67

using 关键字可能需要 3 个输入列 , 用来设置坐标。

plot 的 thru smooth 关键字不能用于 splot。splot 使用 cntrparam 和 dgrid3d 关键字 完成平滑功能。

单个空白行用来分割数据块。连续两个空白行用于分割数据集。一次 `splot` 会读取一个数据集中的所有数据块，但是数据块之间不会使用线条连接，但是数据块内的点会根据样式的需求连接。如果所有数据块都拥有相同的行数，`gnuplot` 会在数据块之间绘制交叉等值线，它会连接相对应的点，形成网格。通常它们被称作网格数据。

Matrix

`Gnuplot` 使用两种方式输入矩阵：

1) `gnuplot` 数据 `x y` 坐标分别是行号和列号，并且把矩阵的值放到指定的坐标上。对于文本数据这是默认的处理方式。

实例：

```
splot 'file' matrix using 1:2:3 #文本数据
splot 'file' binary general using 1:2:3 #二进制数据
```

自动产生 `x` 和 `y`, i. e.,

`z11 z12 z13 z14 ...`

`z21 z22 z23 z24 ...`

`z31 z32 z33 z34 ...`

因此 输入列 1 2 分别代表自动产生的 `x` 和 `y`，输入列 3 代表矩阵中的 `z` 值。

对于文本输入，一个空白行和一个注释行，都会结束一个矩阵的读取。但要注意，`gnuplot` 在绘制之前会检查整个数据块的维度。如果某个数据块不是方阵，那么会提示 "Matrix does not represent a grid"。所以你不能从夹杂着非矩阵数据的文件中读取矩阵。

2)

第二种矩阵格式是非均匀矩阵。使用 `nonuniform` 关键字。

```
splot 'file' nonuniform matrix using 1:2:3 # 文本输入
splot 'file' binary matrix using 1:2:3 ##二进制输入
```

输入矩阵的格式为：

`<N+1> <y0> <y1> <y2> ... <yN>`

`<x0> <z0,0> <z0,1> <z0,2> ... <z0,N>`

`<x1> <z1,0> <z1,1> <z1,2> ... <z1,N>`

:

很明显以上矩阵格式实际上是 x y 对应 z 值的表格。它会产生如下点：

<x0> <y0> <z0,0>

<x0> <y1> <z0,1>

<x0> <y2> <z0,2>

...

<x1> <y0> <z1,0>

<x1> <y1> <z1,1>

<x1> <y2> <z1,2>

... ..

因此 输入列 1 2 分别代表自动产生的 x 和 y，输入列 3 代表矩阵中的 z 值。

处理矩阵和数组的 C 函数在 binary.c 里面，函数原形：

```
int fwrite_matrix(file, m, nrl, nrl, ncl, nch, row_title, column_title)
```

函数使用实例在 bf_test.c 它可以产生由 binary.dem 使用的二进制数据文件。

实例：

在文本模式下调整矩阵位置：

```
splot 'dat' matrix using (1+$1):(1+$2*10):3
```

绘制矩阵的第 3 行。

```
plot 'dat' matrix using 1:3 evvery 1:999:1:2
```

(行索引从 0 开始，因此第三行为 2)

使用 binary 关键字可以从二进制数据中读取数据。参见 plot binary 67

数据文件实例

```
splot 'data.dat'
```

data.dat 包含

```
# The valley of the Gnu.
0 0 10
0 1 10
0 2 10

1 0 10
1 1 5
1 2 10
```

```
2 0 10
2 1 1
2 2 10

3 0 10
3 1 0
3 2 10
```

数据文件是 4*3 网格数据。

注意网格数据每个采样行的 x 值保持恒定。你也可以保持 y 值恒定。当各个数据块的行数相同，数据就被认为是网格数据。如果你的数据不规律(x y 都不恒定)，网格的连接会非常混乱。

网格数据

3d 程序使用采样点，数据点绘制网格数据的网格顶点。当数据来自函数时，使用 set isosamples 控制采样。在函数绘制中，samples 选项控制采样，它和 isosamples 不一样：因为并不是每个 x 等值采样线都和 y 等值采样线相交。在处理数据文件时，如果每个数据块的点数目相同。那么 gnuplot 会连接相邻数据块的对应点，形成交叉的等值线，因而会产生网格。如果开启 contour 选项或者 hidden3d 选项图形会产生等高线和遮蔽效果。

等高线算法，会测试 z 值密度。它比较某个 x 等值线上相邻 y 采样点的 z 值。因此如果一个 x 等值线上的等高线采样点不和某个 y 等值线匹配，那么这个采样点会被忽略。

试试看：

```
set xrange [-pi/2:pi/2]; set yrange [-pi/2:pi/2]
set style function lp
set contour
set isosamples 10,10; set samples 10,10;
splot cos(x)*cos(y)
set samples 4,10; replot
set samples 10,4; replot
```

绘制曲面

splot 可以通过连接数据点而产生曲面网络。这些点可以从数据文件中读取，也可以通过函数进行自动采样。采样点的密度受到 set isosample，set sample 控制。参考

set surface：显示曲面

set hidden3d，开启曲面的 3d 遮蔽

set view：3d 视角

曲面可以通过用线段直接连接数据点得到，也可以通过平滑曲线逼近数据点得到（set cntrparam）。

stats 统计命令

语法：

```
stats ' filename' [using N[:M]] [name ' prefix' ] [[no]output]]
```

这个命令为数据文件中的某 1 列和 2 列准备统计信息。using every index 关键字和 plot 命令用法相同。统计信息默认输出在屏幕上，可以使用 set print 命令重定向。使用 nooutput 关键字关闭输出，只影响内置变量。

```
STATS_records
# 在取值范围内的记录总数。
STATS_outofrange
# 超出取值范围的记录总数
STATS_invalid
# 所有错误的记录总数：数据不完整，丢失，数据错误。
STATS_blank
# 文件中空白行的数目
STATS_blocks
# 可索引的数据块总数，参考 plot index 关键字 74
```

如果指定的坐标轴开启了自动缩放，可能不会有超出取值范围的数据。如果 stats 命令同时分析 2 个输入列，那么内置变量名在第一输入列后缀_x，第二输入列后缀_y。例如：STATS_min_x 和 STATS_min_y。

```
STATS_min # 所有取值范围以内的数据最小值。
STATS_max # 所有取值范围以内的数据最大值
STATS_index_min # 索引 i 使得，data[i] == STATS_min
STATS_index_max # 索引 i 使得，data[i] == STATS_max
STATS_lo_quartile # 第一四分位值
STATS_median # 中值
STATS_up_quartile # 第三四分位值
STATS_mean # 所有取值范围内数据的主值
STATS_stddev # 取值范围内数据的标准差
STATS_sum # 和数据之和
STATS_sumsq # 平方和
```

以下变量只在 2 列输入列时有效：

```
STATS_correlation # x 和 y 值的关联系数
STATS_slope # 线性拟合  $y = Ax + B$  的 A 值
STATS_intercept # 线性拟合  $y = Ax + B$  的 B 值
STATS_sumxy # x*y 的和
STATS_pos_min_y # 取得最小 y 值的 x 值
```

```
STATS_pos_max_y # 取得最大 y 值的 x 值
```

使用 **name** 关键字可以指定变量前缀（默认是 **STATS**）。这在统计多个文件时可以方便的取回数据。

```
stats "dat1" using 2 name 'A'
stats "dat2" using 2 name 'B'
if(A_mean<B_mean){ }
```

STATS_index_XX 得到的索引值，和 **plot** 命令的第 0 数据列对应(\$0)。第一个点索引为 0，第 N 个点索引为 N-1。

为了寻找中值和四分位点 数据会被排序。如果数据总数 N 是奇数，中值等于排序后索引为(N+1)/2 的数，如果 N 是偶数中值等于排序后索引为 N/2 和(N+1)/2 的数的算数平均数。

参考实例：**stats.dem**

当前版本的 **gnuplot** 不允许对同时开启 x 和 y 轴自动缩放的数据进行统计。在以后版本的 **gnuplot** 宏可能会解决这个问题。

System

system "cmd" 用来通过 shell 调用系统命令。参考 shell 命令。通过函数的方法调用：**res = system ("cmd")** 会返回 cmd 的标准输出到变量 **res**。

这可以用于对 **gnuplot** 的扩展，比如你写了一个 **awk** 脚本用来根据原始数据产生适合绘制的文件。你希望 **gnuplot** 绘图时自动调用这个脚本。可以如下：

```
# For linux/unix
datafile="mydat.dat"
tmpfile=system("mktemp gnuplot_XXXXX_tmp")
# 利用 shell 产生临时文件
system(sprintf("awk -f file.awk %s > %s", datafile, tmpfile))
# 调用 awk 输出到临时文件中
plot tmpfile #绘制临时文件的数据
system("rm -f gnuplot*_tmp")
# 删除临时文件
```

test

Test 命令用于输出内置的绘制样式和调色板的实例到当前输出终端。

语法：

```
test {terminal | palette [rgb|rbg|grb|gbr|brg|bgr]}
```

test terminal 显示 线条和点的样式。

Test palette 绘制调色板的 RGB 测试。

Undefine

用来删除若干用户变量。可用于重置脚本。

参数变量名的最后一个字符可以是星号 *，它是通配符。并且通配符只能出现在变量名末尾。

实例：

```
undefine foo foo1 foo2 #删除 3 个变量
bar=1;bar1=2;bar2=3
undefine bar* #删除所有以 bar 开始的变量
```

unset

是 set 命令的反命令。set 用于开启某个选项。unset 用于关闭某个选项。unset 命令可以包含迭代语句，参见：迭代 63 和 85。

实例：

```
set xtics mirror
unset xtics
unset for [i=100:200] label i
```

update

此命令将当前拟合的参数写入指定文件。此文件是合法的 gnuplot 脚本。

语法：

```
update <filename> {<filename>}
```

如果第二个文件名存在，那么使用第二个文件，保持第一个不变。

如果待写入的文件存在，原始文件会被重命名加上后缀 .old。在 dos 系统中由于文件名长度存在限制，因此 filename.ext 会被重新命名为 filename.old 并且 filename 可能转换成短文件名。在 VMS 系统中不会发生重命名，因为他有文件版本管理机制。

参考 fit 命令 58。

while

语法：

```
while (<expr>) {
<commands>
}
```

如果<expr>是非零，那么就被当作 true。这个命令不能与旧的 if 格式混合使用，参考：旧 if 63。

第四部分

常用的终端列表

gnuplot 支持大量的输出格式，可以使用 `set terminal` 选择终端类型。

完整的终端列表查看英文手册。这里只列出常用的。大部分情况你只需要了解 `eps` , `latex` , `png` , `jpeg` , `gif` 终端。这些终端支持 `size` 参数控制画布大小，`font` 参数控制文字字体。由于 `png` , `jpeg` , `gif` 终端使用外部库来生成文件，由于库不一定支持多字节文字（中文），因此你的输出可能中文乱码。此时尝试使用 `eps` 或者 `latex`。

另外有意思的终端是：`Dumb` 终端用来生成字符画。

Aqua

这个驱动依赖于 `AquaTerm.app` 。它用于在 Mac OS X 上显示。

Syntax:

```
set terminal aqua {<n>} {title "<wintitle>"} {size <x> <y>}  
    {font "<fontname>{,<fontsize>}" }  
    {{no}enhanced} {solid|dashed} {dl <dashlength>}}
```

where `<n>` is the number of the window to draw in (default is 0), `<wintitle>` is the name shown in the title bar (default "Figure <n>"), `<x>` `<y>` is the size of the plot (default is 846x594 pt = 11.75x8.25 in).

Use `<fontname>` to specify the font (default is "Times-Roman"), and `<fontsize>` to specify the font size (default is 14.0 pt).

The aqua terminal supports enhanced text mode (see enhanced (p. 23)), except for overprint. Font support is limited to the fonts available on the system. Character encoding can be selected by `set encoding` and currently supports iso latin 1, iso latin 2, cp1250, and UTF8 (default). Lines can be drawn either solid or dashed, (default is solid) and the dash spacing can be modified by `<dashlength>` which is a multiplier > 0.

Canvas

canvas 终端产生一个 javascript，它可以用在 HTML5 的画布对象。

详情参见英文手册。

Cairolatex

Cairolatex 终使用 `cairo` 和 `pango` 支持库产生封装的 PostScript (`.eps`) 文件或 pdf 文件。然后使

用 epslatex 类似的 latex 程序产生文本输出。

Cgi

Cgi 和 hcgi 用来支持 SCO CGI 打印机。Hcgi 用于打印机。

Cgm

Cgm 用来生成 Computer Graphics Metafile Version1 .这个文件格式是 ANSI X3 122-1986 的子集。

详情参见英文手册。

Contex

ConTex 是 Tex 的一个宏包。它与 Metapost 高度整合。可以产生高质量的 PDF 文件。

详情参见英文手册。

Corel

Corel 驱动用来支持 CorelDraw

详情参见英文手册。

Debug

这个驱动用来调试 gnuplot。它对于想修改 gnuplot 源代码的人有用。

Dumb

Dumb 终端驱动使用 ASCII 文本产生字符画。

语法：

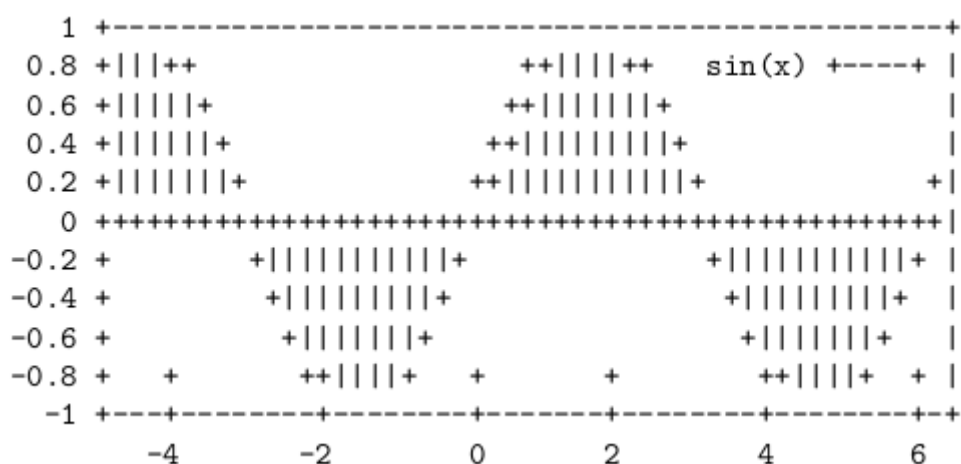
```
set terminal dumb {size <xchars>,<ychars>} {[no]feed}
                {[no]enhanced}
```

<xchars> 和 <ychars> 表示产生的字符画的尺寸，默认是 79*24.

图形结尾的换行只有使用 feed 选项才输出。

实例：

```
set term dumb size 60,15
plot [-5:6.5] sin(x) with impulse
```



dxg

Dxg 终端用于产生可在 AutoCad 10.X 中应用的图形。

详情参阅英文手册。

Dxy800a

用于 Roland DXY800A 绘图机。

Eepic

Eepic 用于支持扩展的 Latex 图形环境。它是 latex 驱动替代品。

详情参阅英文手册。

Emf

Emf 产生 Enhanced Metafile Format 文件。这种格式的文件被大多数 windows 程序识别。

详情参阅英文手册。

Emxvga

Emxvga, emxvesa, vga 终端驱动用来支持 PC 的 SVGA vesa SVGA VGA 显卡。

详情参阅英文手册。

Epscairo

Epscairo 终端产生封装的 PostScript 文件。它使用 cairo 和 pango 支持库。Cairo 版本 ≥ 1.6

详情参阅英文手册。

Epslatex

Epslatex 产生 latex 脚本，它可以被 latex 载入。

详情参阅英文手册。

Epson_180dpi

epson_180dpi 和 epson_60dpi 驱动用来支持 Epson LQ-style 24-pin 打印机。

fig

Fig 终端驱动用来产生使用 fig 图形语言描述的文件。

详情参阅英文手册

ggi

Ggi 驱动可以运行在 X 和 svgalib 不同的目标环境下。

详情参阅英文手册

gif

产生 gif 图形。

详情参阅英文手册

jpeg

产生 jpeg 文件

详情参阅英文手册

latex

产生 latex 脚本。

详情参阅英文手册

mp

产生 metapost 程序能够处理的文件。

详情参阅英文手册

pbm

产生 pbm 图形文件。

详情参阅英文手册

pdf

产生标准 pdf 文件

详情参阅英文手册

pdfcairo

pdfcairo 终端驱动用于产生 pdf 文件，但是文件 2d 图形使用 cairo 产生，文本使用 pango 产生。

详情参阅英文手册

png

产生 png 图形文件

详情参阅英文手册

pslatex 和 pstex

用于生成 ps 格式的文件。pslatex 产生的文件可以被 latex 载入，pstex 可以被 tex 载入。Pslatex 产生的文件可以被 dvips 和 xdvi 识别。

详情参阅英文手册

pstricks

Pstricks 产生使用 latex 宏包：pstricks.sty。他是 eepic 和 latex 驱动替代品。

详情参阅英文手册

Qt

qt 驱动使用 qt 库，把图形输出到一个独立的 QT 窗体中。

详情参阅英文手册

tpic

tpie 终端驱产生 latex 文件，支持使用 tpic 的 latex 图形环境。

详情参阅英文手册

wxt

将图形输出到可交互的独立窗体。窗体使用 wxWidgets 创建。

详情参阅英文手册

X11

在 X11 环境下产生交互式窗体，并且把图形输出到窗体中。Persist 属性在会话结束以后保留窗体不关闭。

详情参阅英文手册