

# AlterMundus



著：Alain Matthes

译：耿楠(陕西·杨凌)

January 3, 2021 Documentation V.1.4c

<http://altermundus.fr>

# tkz-fct

AlterMundus

Alain Matthes

**tkz-fct.sty(v1.4c)** 是一个基于 **gnuplot** 和 **TikZ** 设计的用于绘制二维函数曲线的宏包，它语法简单、使用方便，是 **tkz**-系列宏包的一个模块，主要用于绘制数学教学中的各类函数曲线，这些函数曲线符合法国高中教学的需求。

☞ 在此，要感谢 **Till Tantau** 开发了精彩的**TikZ**宏包，也要感谢 **Michel Bovani** 开发的**fourier**宏包，这些宏包都能与**utopia**宏包很好的协同工作。

☞ 感谢 **David Arnold**，他纠正了宏包中许多错误，并对多数示例进行了测试；感谢 **Pablo González Luengo**，他在 Github 仓库的文档管理方面提供全力的帮助；感谢 **Wolfgang Büchel**，他也纠正了许多错误，并编写优秀的脚本管理示例文件；感谢 **John Kitzmiller**，他设计并测试了更多的示例；最后要感谢 **Gaetan Marris**，他的评论为宏包的改进提供了很好的思路。

☞ 如果发现该文档的错误或有其他任何意见和建议，请发信至：[Alain Matthes](#)。

☞ 如果发现译文的错误或有其他任何意见和建议，请发信至：[耿楠](#)。

可以在[CTAN](#)发布的“LATEX Project Public License”协议下发布和修改该文档。

## Contents

1	概述	6
2	使用 <code>gnuplot</code> 绘图	7
2.1	TikZ 与 <code>gnuplot</code> 协作绘图	7
2.2	安装 <code>gnuplot</code>	9
2.3	测试 <code>tkz-base</code>	9
2.4	测试 <code>tkz-fct</code>	9
3	安装 <code>tkz-fct</code> 宏包	11
3.1	在 OS X、Linux 和 Windows 中基于 TeXLive 发行版使用 <code>tkz-fct</code> 宏包	11
3.2	在 Windows 中基于 MikTeX 使用 <code>tkz-fct</code>	12
3.3	安装小结	12
4	命令简介	14
4.1	<code>\tkzFct</code> 命令：用 <code>gnuplot</code> 绘制函数曲线	14
4.2	<code>samples</code> 选项	14
4.3	<code>xstep</code> , <code>ystep</code> 选项	15
4.4	改变 <code>xstep</code> 和 <code>ystep</code> 选项	15
4.5	<code>ystep</code> 选项和常数函数	16
4.6	仿射函数或线性函数	17
4.7	子风格	17
4.8	使用 <code>tkz-base</code> 宏包提供的命令	17
5	<code>\tkzDefPointByFct</code> 命令：通过函数定义点	19
5.1	<code>\tkzGetPoint</code> 命令示例	19
5.2	<code>\tkzGetPoint</code> 和 <code>tkzPointResult</code> 命令示例	20
5.3	<code>draw</code> 和 <code>ref</code> 选项	20
5.4	仅绘制函数曲线上的点	21
5.5	与实际坐标无关的点	22
5.6	设置点的样式	23
6	标注	24
6.1	添加标注示例	24
7	切线绘制命令	25
7.1	<code>\tkzDrawTangentLine</code> 命令：绘制函数切线	25
7.2	<code>xstep</code> 和 <code>ystep</code> 不为 1 的切线	25
7.3	<code>kl</code> 、 <code>kr</code> 和 <code>draw</code> 选项	26
7.4	<code>with</code> 选项	27
7.5	切线示例	28
7.6	半切线	28
7.7	Lorentz 曲线的半切线	29
7.8	切线族示例	31
7.9	无曲线的切线族	31
7.9.1	<code>\tkzFctLast</code> 宏	32
7.10	计算的历史记录	33
7.10.1	提前定义计算结果	33
7.10.2	使用计算结果	33
8	区域填充命令	34
8.1	<code>\tkzDrawArea</code> 或 <code>\tkzArea</code> 命令：区域填充	34
8.2	对数函数	34
8.3	简单示例	35

8.4	填充样式	37
8.5	<code>\tkzDrawAreafg</code> 命令: 填充两条函数曲线之间的区域	38
8.6	两条函数曲线之间的区域	38
8.7	设置填充图案	39
8.8	<code>between</code> 选项	39
8.9	两条 Lorentz 函数曲线之间的区域	40
8.10	混合样式	41
8.11	水平曲线	42
9	黎曼和	43
9.1	求黎曼和	43
9.2	<code>\tkzDrawRiemannSumInf</code> 命令示例	44
9.3	<code>\tkzDrawRiemannSumSup</code> 命令示例	45
9.4	<code>\tkzDrawRiemannSumMid</code> 命令示例	46
10	特殊直线	47
10.1	垂线	47
10.2	绘制垂线	47
10.3	绘制多条垂线	48
10.4	用 <code>fp</code> 计算垂线	48
10.5	水平线	49
10.6	水平渐近线	49
10.7	绘制多条水平线	50
10.8	水平和垂直渐近线	51
11	参数方程的函数曲线	52
11.1	参数曲线示例 1	52
11.2	参数曲线示例 2	53
11.3	参数曲线示例 3	54
11.4	参数曲线示例 4	55
11.5	参数曲线示例 5	56
11.6	参数曲线示例 6	57
11.7	参数曲线示例 7	58
12	极坐标函数曲线	59
12.1	极坐标函数曲线示例 1	59
12.2	极坐标函数曲线示例 2	60
12.3	极坐标心形	61
12.4	极坐标函数曲线示例 4	61
12.5	极坐标大麻曲线	63
12.6	斯卡贝斯曲线	64
13	标记符号	65
14	部分实例	66
14.1	TikZ + <code>tkz-fct</code> 实例	66
14.2	Lorentz 曲线	67
14.3	指数曲线	68
14.4	对数坐标轴	69
14.5	两条函数曲线	70
14.6	函数插值	71
14.6.1	Le code	71
14.6.2	插值曲线	71

14.7 Van der Waals曲线 . . . . .	73
14.7.1 参数关系表 . . . . .	73
14.7.2 $b=1$ 的第一曲线 . . . . .	74
14.7.3 $b=1/3$ 的第二曲线 . . . . .	75
14.7.4 $b=32/27$ 的第3曲线 . . . . .	76
14.8 临界值 . . . . .	77
14.8.1 Van der Waals曲线 . . . . .	77
14.8.2 Van der Waals曲线(续) . . . . .	78
15 在alterqcm和tkz-tab宏包中绘制函数曲线 . . . . .	79
15.0.1 第1种方式 . . . . .	80
15.0.2 第2种方式 . . . . .	81
16 pgfmath和fp.sty工具 . . . . .	83
16.1 pgfmath . . . . .	83
16.2 fp.sty . . . . .	83
17 注意事项 . . . . .	85
17.1 gnuplot的函数 . . . . .	86
18 命令列表 . . . . .	87
18.1 tkz-fct宏包提供的所有命令 . . . . .	87
18.2 tkz-base宏包提供的命令 . . . . .	87
Index . . . . .	88

## 1 概述

TikZ 为绘制函数曲线提供了多种方式,但其`pgfmath`库太慢,且结果精确不高,因此,使用`gnuplot`工具绘制函数曲线是一个更为合理的选择。

在 TikZ 中,使用`gnuplot`工具<sup>1</sup>绘制函数曲线的基本语法是:

```
\draw[options] plot function {gnuplot 函数表达式};
```

在新版`tkz-fct`宏包中,用`\tkzFct`命令及 TikZ 选项进行绘图。其中,最重要选项是`domain`和`samples`。

本宏包使用`\tkzFct`命令代替了`\draw plot function`,同时,还实现了两个重要的操作。首先是用`gnuplot`语法保存函数表达式,同时,也用`fp`保存该函数表达式,以方便后续使用。这使得可以在不返回表达式的情况下,在曲线上布置点(通过`fp`计算),或者绘制切线。

当然,最重要的是,`\tkzFct`命令能独立使用横坐标和纵坐标步长单位,其步长单位由`tkz-base`宏包的`\tkzInit`命令,通过`xstep`和`ystep`选项进行设置。

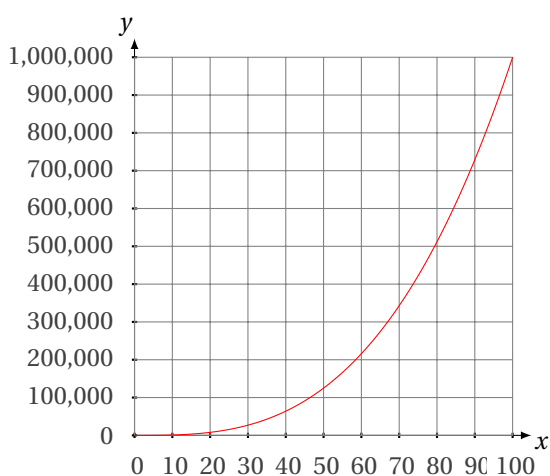
`\tkzFct`命令可以使用`domain`选项和函数表达式。如果`xstep`和`ystep`的步长为 1,则直接使用值域范围和计算结果绘制图像。如果`xstep`的步长不为 1,则表达式中的自变量须使用`\x`宏表示,这能有效避免计算中的数据溢出问题。

例如,有如下函数 $f$ :

$$f(x) = x^3 \text{ 其中 } 0 \leq x \leq 100$$

$f(x)$  的值域在 0 到 1,000,000 之间,令`xstep=10`和`ystep=100000`,则坐标轴长度为 10 cm(无缩放)。

这可以将值域限定在 0 到 10 之间,但`gnuplot`表达式中的`\x`则相当于 $x \times 10$ 。将计算结果除以`ystep=100000`,可以使 $f(x)$ 的结果不变,仍在 0 到 10 之间。



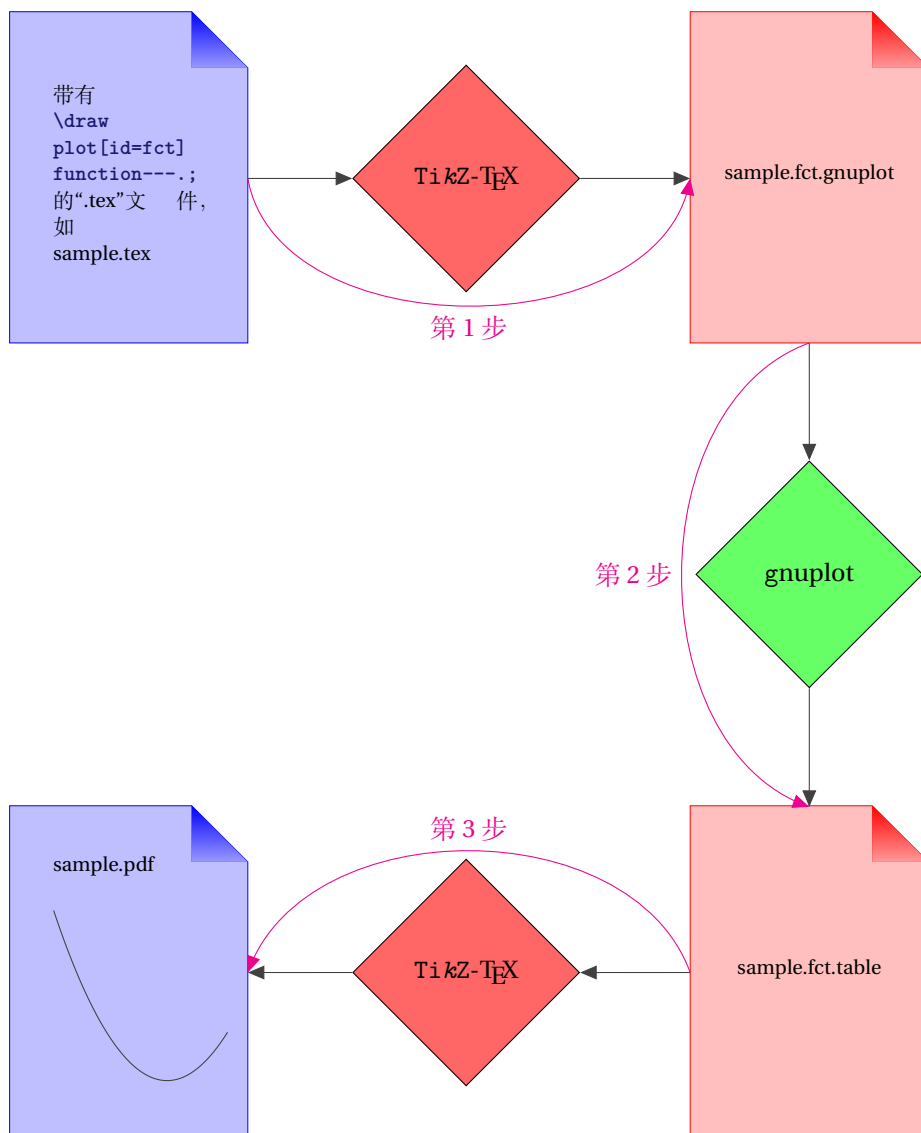
```
\begin{tikzpicture}[scale=.5]
  \tkzInit[xmax=100,xstep=10,
    ymax=1000000,
    ystep=100000]
  \tkzDrawX[right]
  \tkzDrawY[above]
  \tkzLabelX[below=6pt]
  \tkzLabelY[left=6pt]
  \tkzGrid
  \tkzFct[color=red,
    domain=0:100]{\x**3}
\end{tikzpicture}
```

<sup>1</sup> `gnuplot` 是一个交互式命令行绘图工具,需要提前安装该软件。

## 2 使用 `gnuplot` 绘图

### 2.1 TikZ 与 `gnuplot` 协作绘图

$\text{T}_{\text{E}}\text{X}$  本质上是一个文本处理系统，虽然可以用  $\text{T}_{\text{E}}\text{X}$  进行计算，但其方法有限。同样，受  $\text{T}_{\text{E}}\text{X}$  制约，TikZ 的计算功能也不够完善。由于  $\text{T}_{\text{E}}\text{X}$  的尺寸范围  $=\pm 16383.99999\text{pt}$ ，而  $1\text{cm}=28.45274\text{pt}$ ，所以  $\text{T}_{\text{E}}\text{X}$  只能处理小于 5.75 厘米范围内的数据。看起来，这已经足够大了，但在一系列的计算中，却很容易超出这一限制。



简单地借助`gnuplot`工具, `TikZ` 就可以绕过这些限制, 从而更加方便地绘制二维图像。

`tkz-fct.sty`宏包基于`gnuplot`工具和`fp.sty`宏包, 用`gnuplot`获取需要绘图点集, 用`fp.sty`宏包实现计算。

因此, 必须安装`gnuplot`工具, 另外`TEX`发行版要有访问`gnuplot`工具的权限, 同时, `TEX`也应该允许`gnuplot`能够写入需要的文件。关于如何安装`gnuplot`工具, 请查阅相关资料, 在此不再冗述。

### • 第1步

例如`sample.tex`文件中的代码如下:

```
\documentclass{article}
\usepackage{tikz}
\begin{document}
\begin{tikzpicture}
\draw plot[id=f1,samples=200,domain=-2:2] function{x*x};
\end{tikzpicture}
\end{document}
```

编译这一段`TikZ`代码, 会生成一个`sample.f1.gnuplot`文件, 该文件的文件名由当前工作文件名`\jobname`和`id`选项值组合而成。因此, 一个文件可能会生成多个不同的`.gnuplot`文件。以`gnuplot`为扩展名的文件是一个普通的纯文本文件, 该文件中包含一条命令, 用于启用`gnuplot`工具创建坐标点数据表文件, 并且为坐标点数据表文件命名, 设置坐标精度。同时, 还用`samples`选项设置了用 $x \rightarrow x^2$ 定义的函数的采样点个数, 当然, 也可以手动编写该`gnuplot`文件:

```
set table; set output "sample.f1.table"; set format "%.5f"
set samples 200; plot [x=-2:2] x*x
```

这两条命令能够创建一个点坐标构成的数据表, 并保存到“`sample.f1.table`”文件中, 本例中, 坐标包含5位小数, 共有201个采样点。

### • 第2步

由于必须用`gnuplot`工具打开`sample.f1.gnuplot`, 因此, 一方面`TEX`要允许`gnuplot`打开`sample.f1.gnuplot`<sup>2</sup>, 另一方面`TEX`要能找到`gnuplot`工具<sup>3</sup>。

如果`gnuplot`能找到`sample.f1.gnuplot`文件, 并且其函数表达式正确, 则会生成`sample.f1.table`坐标点数据表文件。

### • 第3步

`TikZ` 使用`sample.f1.table`数据表中的坐标数据绘图。

```
# Curve 0 of 1, 201 points
# Curve title: "x*x"
# x y type
-2.00000 4.00000 i
-1.98000 3.92040 i
-1.96000 3.84160 i
---.
1.98000 3.92040 i
2.00000 4.00000 i
```

1. 注意如果`sample.f1.gnuplot`时间戳没有变化, 则不会再次启动`gnuplot`工具, 并且仍然使用当前`sample.f1.table`数据表中的坐标数据进行绘图。

<sup>2</sup> 请参阅`TEX`的`--shell-escape`和`--enable-write18`命令行参数。

<sup>3</sup> 需要设置正确的`PATH`环境变量。



2. 如果需要手动运行`gnuplot`，那么第一次编译后会创建`sample.f1.table`坐标点数据表文件，并在后续处理中使用这个`sample.f1.table`数据表文件。
3. 也可以手动或采用其它方式创建该数据表文件，然后供 TikZ 使用，如：

```
\draw plot[smooth] file {data.table};
```

## 2.2 安装`gnuplot`

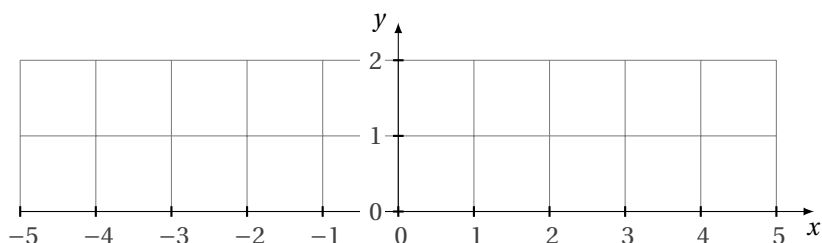
`gnuplot` 是一个跨平台的工具，在多数 Linux 发行版中已安装了该工具。当然，它也适用于 OS X 和 Windows 操作系统。

1. **Ubuntu**：对于 Ubuntu Linux 或其它 Linux 系统，按照常规软件包安装方式安装即可。
2. **Windows**：对于 Windows 则需要注意下载与操作系统匹配的版本，并在安装`gnuplot`后，将“`wgnuplot`”重命名为“`gnuplot`”。然后，修改`path`环境变量，例如：如果程序安装在`C:\gnuplot`目录中，则需要将`C:\gnuplot\bin\`添加到环境变量（选择“计算机”的“属性”，然后在“高级”选项卡中，找到“环境变量”进行添加）。接下来，要注意，在后续的  $\text{\LaTeX}$  编译中，必须使用`--enable-write18` 编译参数进行编译。
3. **OS X**：对于 OS X，则需要从源码编译后进行安装。

## 2.3 测试`tkz-base`

将以下代码保存为“`test.tex`”文件，然后用 `xelatex`、`pdflatex` 或 `lualatex` 进行编译：

```
\documentclass{scrartcl}
\usepackage{tkz-fct}
\begin{document}
  \begin{tikzpicture}
    \tkzInit[xmin=-5,xmax=5,ymax=2]
    \tkzGrid
    \tkzAxeXY
  \end{tikzpicture}
\end{document}
```



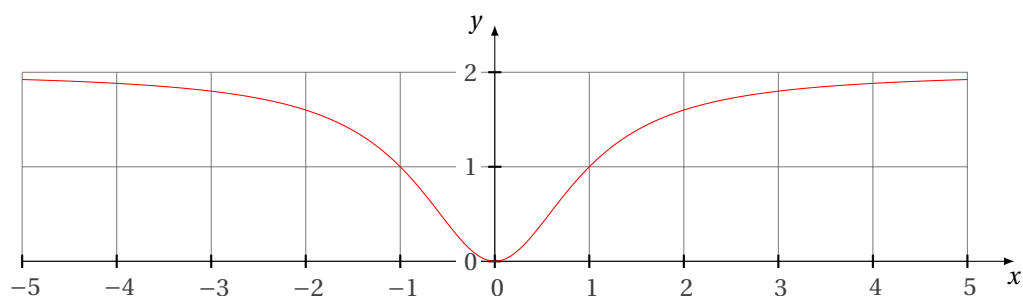
如果能够绘制坐标网格和坐标轴，则表示`tkz-base`宏包已正确安装。

## 2.4 测试`tkz-fct`

仅需在上述代码中添加一句代码就可以绘制函数曲线：

```
\documentclass{scrartcl}
\usepackage[usenames,dvipsnames]{xcolor}
\usepackage{tkz-fct}
\begin{document}
  \begin{tikzpicture}[scale=1.25]
    \tkzInit[xmin=-5,xmax=5,ymax=2]
    \tkzGrid
```

```
\tkzAxeXY
\tkzFct[color=red]{2*x**2/(x**2+1)}
\end{tikzpicture}
\end{document}
```



```
\begin{tikzpicture}[scale=1.25]
\tkzInit[xmin=-5,xmax=5,ymax=2]
\tkzGrid
\tkzAxeXY
\tkzFct[color=red]{2*x**2/(x**2+1)}
\end{tikzpicture}
```

如果能够绘制函数曲线，则表示`tkz-fct`宏包已正确安装。

### 3 安装 `tkz-fct` 宏包

目前，`tkz-fct` 宏包已被收录于CTAN，如果发行版中未收录`tkz-fct`宏包，或需使用该宏包更新版本，或修改后的该宏包的自定义版本，则需要按本节的说明安装`tkz-fct`宏包。注意，由于偶尔会使用`PGF`的 CVS 版本，此处的“`TEXMF`”目录树中包含了`PGF`的文件。

#### 3.1 在 OS X、Linux 和 Windows 中基于 TeXLive 发行版使用 `tkz-fct` 宏包

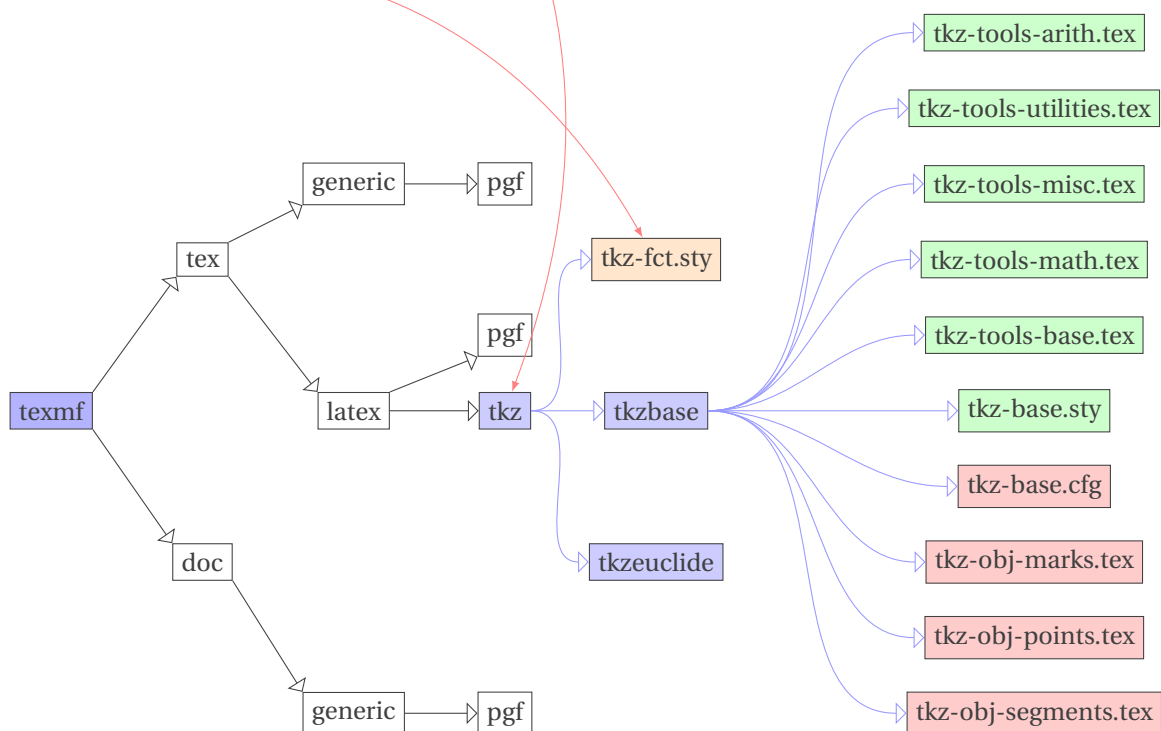
##### TeXLive

在 TeXLive 的目录树中，会创建一个 `tkz` 文件夹，其路径为：`texmf/tex/latex/tkz`。当然，并不强制使用“`tkz`”为文件夹命名，也可以使用其它名称。

`texmf` 是一个文件夹，不同系统其路径可能是不同的，并且也可以自定义其工作路径，例如：

- OS X 系统：`/Users/ego/Library/texmf`；
- Ubuntu 系统：`/home/ego/texmf`。

1. 将 `tkz-fct.sty` 复制到 `tkz` 中。



2. 打开终端，根据需要执行 `sudo texhash` 命令。
3. 检查 `TikZ 2.10` 是否安装，这是确保 `tkz-fct` 宏包正确运行的最低 `TikZ` 版本。同时，也必须先安装 `tkz-base` 宏包。一般来讲，由于比较常用，几乎所在 `TeX` 发行版中都包含了 `fp.sty` 宏包，若没有该宏包，则需要手动安装。

### 3.2 在 Windows 中基于 MikTeX 使用 `tkz-fct`

#### MikTeX Windows

由于对 Windows 系统不熟悉，以下说明中由宏包用户 **Wolfgang Buechel** 提供：

将 `tkz-fct.sty` 宏包安装到 MikTeX<sup>4</sup>：

- 在 `[MiKTeX-dir]/tex/latex` 中添加 `tkz` 文件夹
- 将 `tkz-fct.sty` 及其所需要的文件复制到 `tkz` 文件夹
- 在命令行中执行 `mktexlsr -u` 命令升级 MikTeX  
或在 `Start/Programs/Miktex/Settings/General` 中  
单击 `Refresh FNDB` 按钮进行更新。

### 3.3 安装小结

需要 `TikZ 2.10` 的支持，安装 `tkz-fct` 宏包，可能的一个问题是需要安装 `gnuplot`。如果发行版中不存在 `tkz-fct` 宏包，则需要在 `TeXLive` 目录树中创建一个文件夹，并将 `tkz-base` 及其需要的文件复制到该文件夹中完成安装。

`tkz-fct` 宏包需要的文件有：

- `tkz-fct.sty` 文件
- `tkz-base` 所需要的文件有：
  - `tkz-base.sty` 主文件
  - `tkz-base.cfg` 配置文件
  - `tkz-tools-base.tex`
  - `tkz-tools-arith.tex`
  - `tkz-tools-misc.tex`
  - `tkz-tools-utilities.tex`
  - `tkz-obj-points.tex`
  - `tkz-obj-segments.tex`
  - `tkz-obj-marks.tex`
- `tkz-euclide` 需要的文件有：
  - `tkz-euclide.sty` 主文件
  - `tkz-tools-intersections.tex`

---

<sup>4</sup> 基于 2.7 版本进行测试。

- `tkz-tools-math.tex`
- `tkz-tools-transformations.tex`
- `tkz-lib-symbols.tex` 添加的新符号
- `tkz-obj-lines.tex`
- `tkz-obj-addpoints.tex` 补充的点的定义
- `tkz-obj-circles.tex`
- `tkz-obj-arcs.tex`
- `tkz-obj-angles.tex`
- `tkz-obj-polygons.tex`
- `tkz-obj-sectors.tex`
- `tkz-obj-protractor.tex`

## 4 命令简介

**tkz-fct**宏包通过**gnuplot**确定绘图需要的点，采样点数由**samples**选项确定，在第1个示例中，采样点数取默认值。得到采样点数据表后，用TikZ实现绘图。

### 4.1 \tkzFct命令：用 gnuplot 绘制函数曲线

该命令是一个重要的命令，它能够绘制连续函数的曲线。

**\tkzFct**[< 命令选项>]{<gnuplot 函数表达式>}

需要绘制图像的函数按采用 **gnuplot** 语法进行表达，其中， $x$  自变量，除非 **xstep** 的值为 1，否则应使用  $\backslash x$ 。

参数	示例	说明
gnuplot 函数表达式	$x**3$	** 表示幂运算 ( $\wedge$ )

类似的函数表达式有： $2*x+1$ 、 $3*\log(x)$ 、 $x*\exp(x)$ 、 $x*x*x+x*x+x$  等。

选项可以是所有有效 TikZ 选项

选项	默认值	含义
domain	xmin:xmax	函数定义域
samples	200	采样点数
id	tkzfct	生成的辅助文件标识名称
color	black	线条颜色
line width	1pt	线条宽度
style	solid	线型

- 🔴 如果 **xstep** 不为 1，则自变量  $x$  需要用  $\backslash x$  表示。
- 🔴 注意在使用 **\tkzFct** 命令之前，需要用 **\tkzInit** 初始化绘图参环境。
- 🔴 注意在参数中不能留有空白。

### 4.2 samples选项

注意，绘制直线的函数曲线，仅需要两个点即可，如

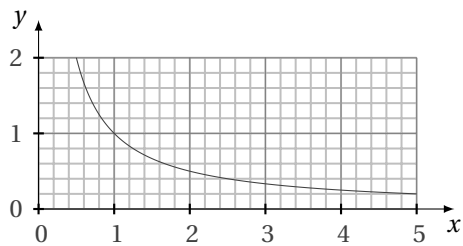
```
\tkzFct[{-(>),color=red,samples=2,domain =-1:2}]{(8-1.5*\x)/2}
```

生成的“xxx.table”数据表中的内容为：

```
# Curve 0 of 1, 2 points
# Curve title: "(8-1.5*x)/2"
# x y type
-1.00000 4.75000 i
2.00000 2.50000 i
```

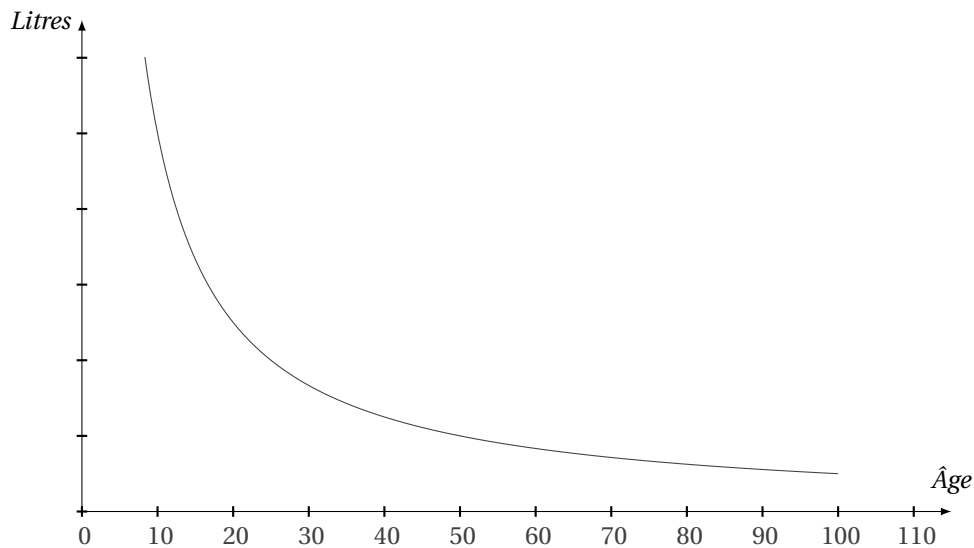
这些数据足够用于绘制简单的线段。

以下代码中，为函数曲线设置了 400 个采样点，**samples** 选项的默认值是 200。



```
\begin{tikzpicture}[scale=1]
  \tkzInit[xmax=5,ymax=2]
  \tkzGrid[sub]
  \tkzAxeXY
  \tkzFct[samples=400,domain=.5:5]{1/x}
\end{tikzpicture}
```

#### 4.3 xstep, ystep选项

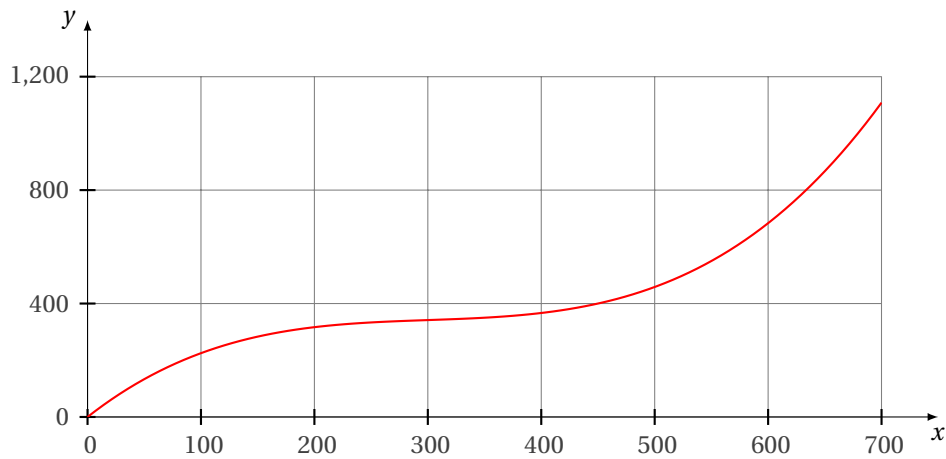


```
\begin{tikzpicture}
  \tkzInit[xmax= 110,xstep=10,
    ymax=6,ystep=1]
  \tkzDrawX[label={\textit{Âge}},below= -18pt]
  \tkzLabelX
  \tkzDrawY[label={\textit{Litres}}]
  \tkzFct[domain = 0.1:100 ]{50/\x}
\end{tikzpicture}
```

#### 4.4 改变xstep和ystep选项

在此，将函数的定义域设置为 0 到 700，并取步长为 **xstep=100**，值域设置为 0 到 2,000。因此，在函数表达式中应该用  $\x$  代替  $x$ 。在 **gnuplot** 中，需要使用类似“1.”和“113.”带小数点的数字进行浮点数除法运算，否则，则会使用整数除法运算。

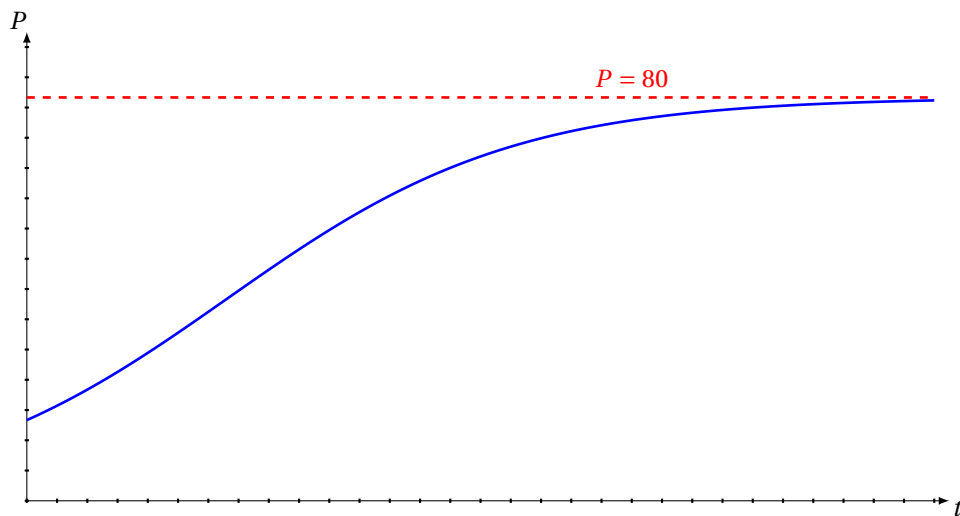
然后，用该命令绘制函数曲线。



```
\begin{tikzpicture}[scale=1.5]
\tkzInit[xmax=700,xstep=100,ymax=1200,ystep=400]
\tkzGrid(0,0)(700,1200) \tkzAxeXY
\tkzFct[color=red,samples=100,line width=0.8pt,domain =0:700]%
    {(1./90000)*\x*\x*\x-(1./100)*\x*\x+(113./36)*\x}
\end{tikzpicture}
```

#### 4.5 ystep选项和常数函数

注意，在此使用`ystep=6`，则`gnuplot`执行 $80 \div 6$ 运算的结果是13。因此，应该使用80.表示常数函数。

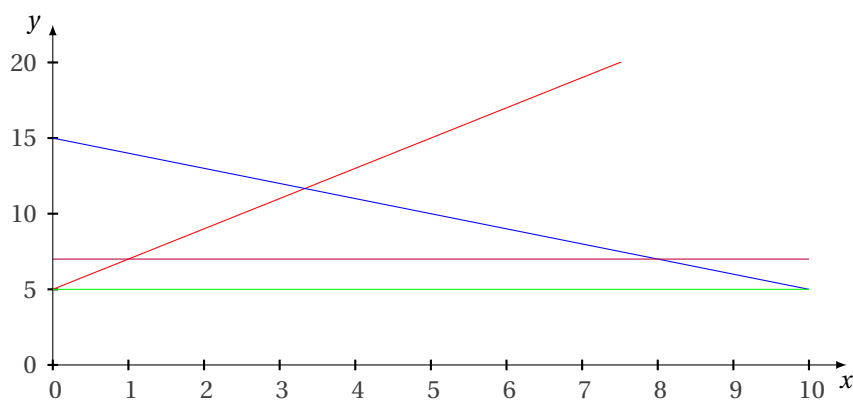


```
\begin{tikzpicture}[scale=0.4]
\tkzInit[xmax=30,ymax=90,ystep=6]
\tkzDrawX[right,label=$t$]
\tkzDrawY[above,label=$P$]
\tkzFct[line width=1pt,color=red,dashed,domain=0:30]{80.0}
\tkzFct[line width=1pt,color=blue,domain=0:30]{80/(1.0+4.0*exp(-0.21*x))}
\tkzText[above,color=red](20,80){$P=80$}
\end{tikzpicture}
```



### 4.6 仿射函数或线性函数

可以使用`gnuplot`绘制直线，虽然这样会增加绘图负担，但为了简化计算，可以只使用 2 个采样点。

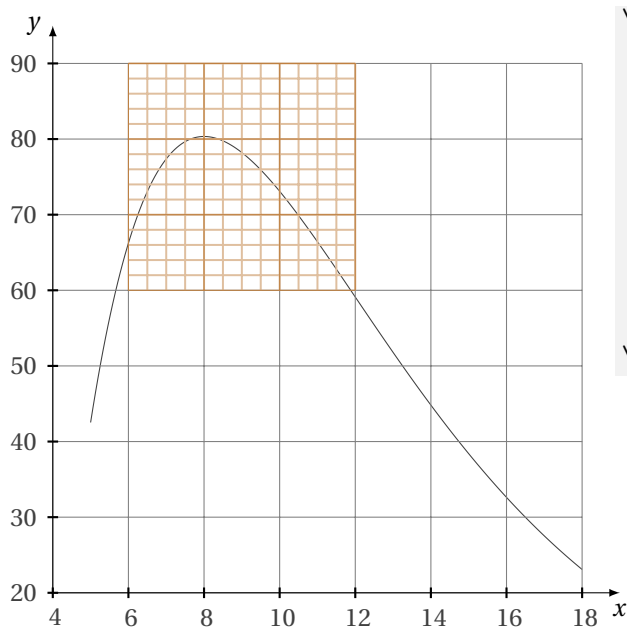


```
\begin{tikzpicture}[]
\tkzInit[ymax=20,ystep=5]
\tkzAxeXY
\tkzFct[color=red,domain=0:10,samples=2]{2*x+5}
\tkzFct[color=blue,domain=0:10,samples=2]{-x+15}
\tkzFct[color=green,domain=0:10,samples=2]{7} % 7/5=1
\tkzFct[color=purple,domain=0:10,samples=2]{7.}%7.0/5 =1.2
\end{tikzpicture}
```

### 4.7 子风格

$$y = (x - 4)e^{-0.25x+5}$$

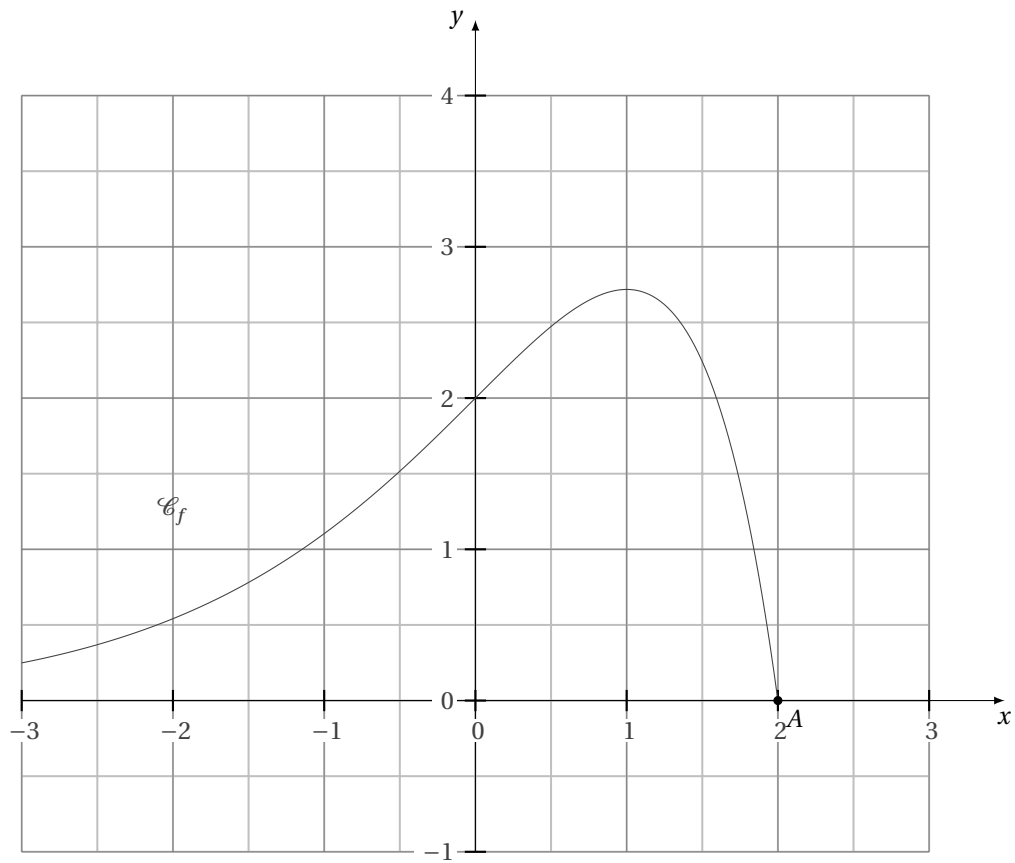
可以采用子网格突出函数曲线的局部。



```
\begin{tikzpicture}
\tkzInit[xmin=4,xmax=18,xstep=2,
          ymin=20,ymax=90,ystep=10]
\tkzFct[domain = 5:18]%
        {(\x-4)*exp(-0.25*\x+5)}
\tkzGrid(4,20)(18,90)
\tkzAxeXY
\tkzGrid[sub,
          subxstep=0.5,
          subystep=2,
          color=brown](6,60)(12,90)
\end{tikzpicture}
```

### 4.8 使用`tkz-base`宏包提供的命令

在绘图过程中，也可以直接使用`tkz-base`宏包提供的各类命令进行绘图。



```
\begin{tikzpicture}[scale=2]
\tkzInit[xmin=-3,xmax=3, ymin=-1,ymax=4]
\tkzGrid[sub,subxstep=.5,subystep=.5]
\tkzAxeXY
\tkzFct[domain = -3:2]{(2-x)*exp(x)}
\tkzText(-2,1.25){$\mathcal{C}_f$}
\tkzDefPoint(2,0){A} \tkzDrawPoint(A) \tkzLabelPoints(A)
\end{tikzpicture}
```

## 5 \tkzDefPointByFct命令：通过函数定义点

**\tkzDefPointByFct(*<decimalnumber>*)**

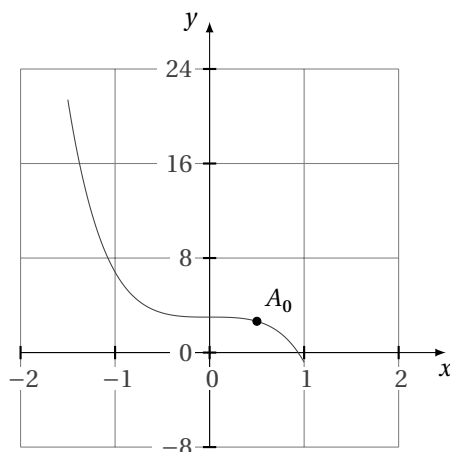
该命令允许使用十进制数表示的  $x$  坐标，通过函数计算，定义函数曲线的一个点。

参数	样例	说明
decimal number	\tkzDefPointByFct(0)	设置横坐标为 0 的函数上的点
选项	默认值	说明
draw	false	允许使用当前样式绘制点
with	a	用指定的函数定义点，函数按自然序用字母编号
ref	empty	设置引用名称

该命令使用最近使用的函数定义点，如果需要使用别的函数，则需要使用旧的 `\tkzFctPt` 命令。可以通过 `\tkzPointResult` 命令使用定义的点，但如果需要绘制该点，则需要使用 `\tkzDrawPoint`。

## 5.1 \tkzGetPoint命令示例

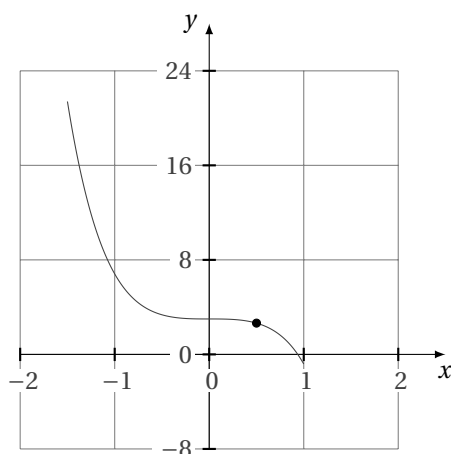
该命令可以引用 `\tkzDefPointByFct` 命令定义的点。



```
\begin{tikzpicture}[scale=1.25]
  \tkzInit[xmin=-2,xmax=2,xstep=1,
           ymin=-8,ymax=24,ystep=8]
  \tkzGrid \tkzAxeXY
  \tkzFct[domain =-1.5:1]{3.0-1.3125*x**5-2.5*x**3}
  \tkzDefPointByFct(.5) \tkzGetPoint{A}\tkzDrawPoint(A)
  \tkzLabelPoint[above right](A){$A_0$}
\end{tikzpicture}
```

## 5.2 \tkzGetPoint和tkzPointResult命令示例

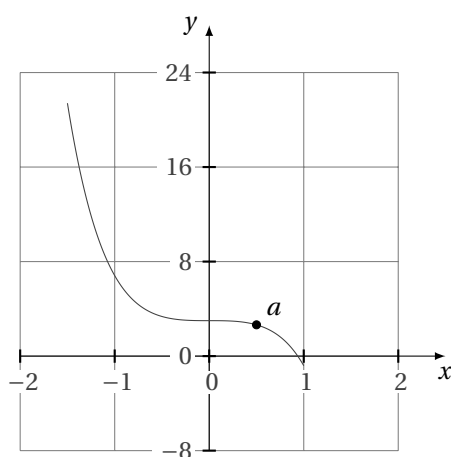
可以直接使用一个定义过点，而不需要保存并命名该点(后续引用)。



```
\begin{tikzpicture}[scale=1.25]
  \tkzInit[xmin=-2,xmax=2,xstep=1,
    ymin=-8,ymax=24,ystep=8]
  \tkzGrid
  \tkzAxeXY
  \tkzFct[domain =-1.5:1]{3.0-1.3125*x**5-2.5*x**3}
  \tkzDefPointByFct(.5)
  \tkzDrawPoint(tkzPointResult)
  % ou bien \tkzDefPointByFct[draw](.5)
\end{tikzpicture}
```

## 5.3 draw和ref选项

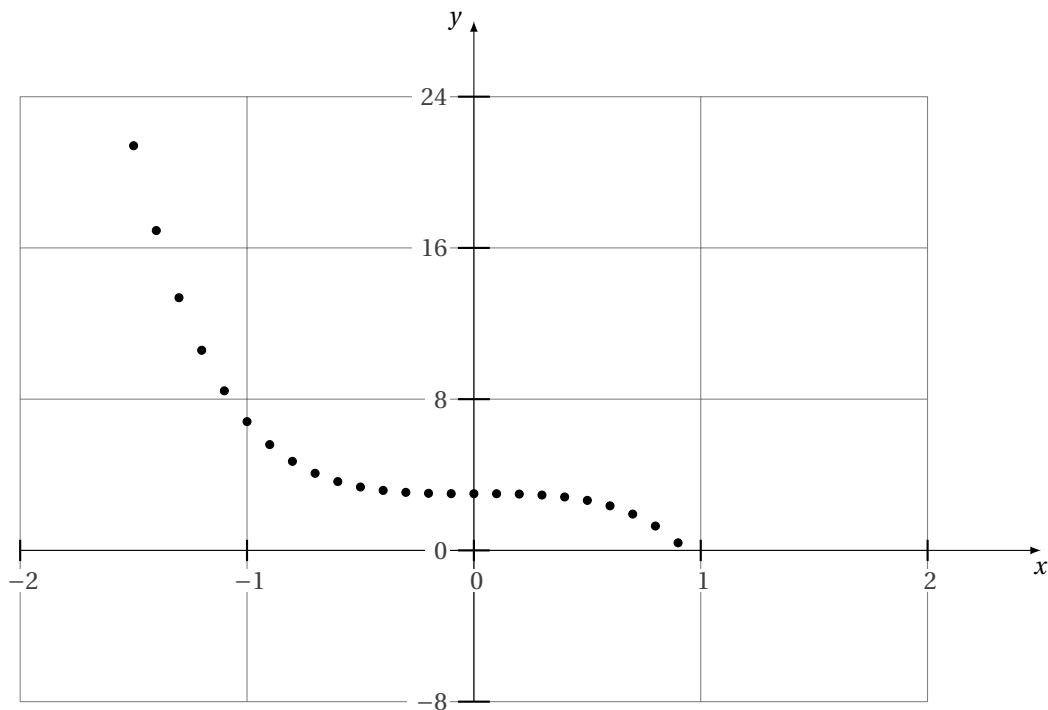
**draw**允许直接绘制定义的点，**ref**允许为定义的点命名，并在后续代码中通过该名称引用该点。



```
\begin{tikzpicture}[scale=1.25]
  \tkzInit[xmin=-2,xmax=2,xstep=1,
    ymin=-8,ymax=24,ystep=8]
  \tkzGrid
  \tkzAxeXY
  \tkzFct[domain =-1.5:1]{3.0-1.3125*x**5-2.5*x**3}
  \tkzDefPointByFct[draw,ref=A](.5)
  \tkzLabelPoint[above right](A){$a$}
\end{tikzpicture}
```

#### 5.4 仅绘制函数曲线上的点

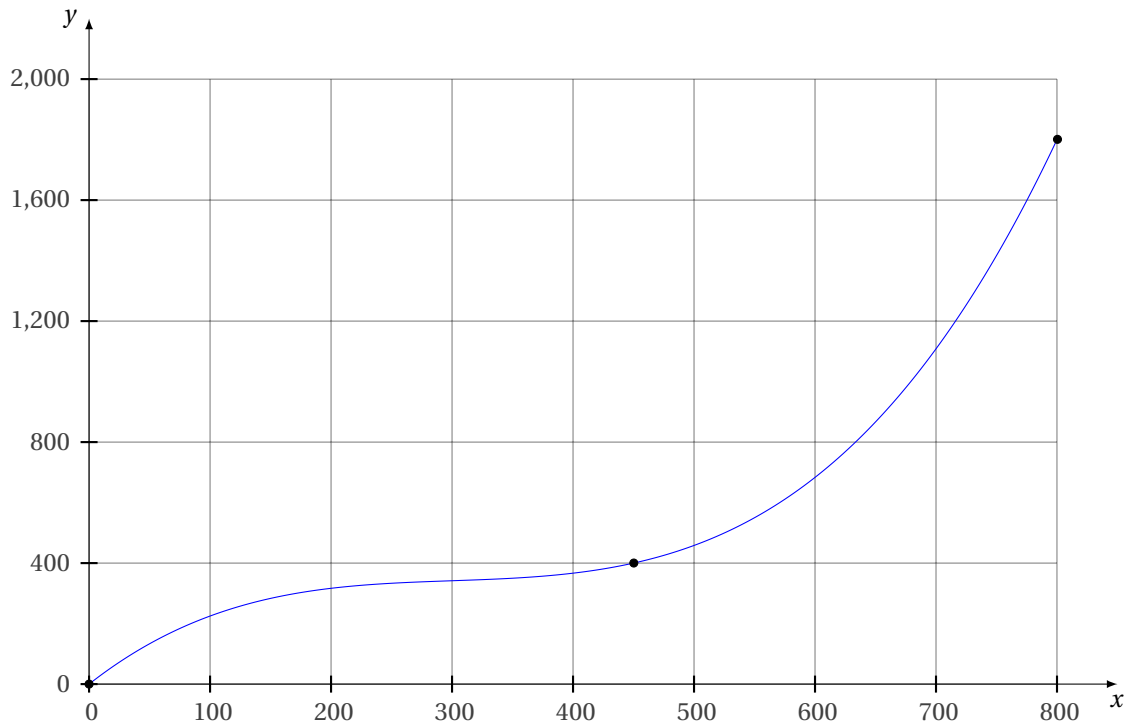
注意，需提前定义表示函数的`\tkzFctLast`全局宏(`\global\edef`)。因为这一操作是通过`fp.sty`宏包实现的，因此，该宏中的函数定义需要使用`fp.sty`的语法。



```
\begin{tikzpicture}[xscale=3,yscale=2]
  \tkzInit[xmin=-2,xmax=2,xstep=1,
    ymin=-8,ymax=24,ystep=8]
  \tkzGrid
  \tkzAxeXY
  \global\edef\tkzFctLast{3.0-1.3125*x^5-2.5*x^3}
  \foreach \va in {-1.5,-1.4,...,1}{%
    \tkzDefPointByFct[draw](\va)}
\end{tikzpicture}
```

### 5.5 与实际坐标无关的点

在此，将函数定义域设置为 0 至 800，值域设置为 0 à 2,000。由于 `xstep=100`，所以需要在函数定义中用 `\x` 代替 `x`。注意用 1. 和 113. 带小数点的数实现浮点数运算。

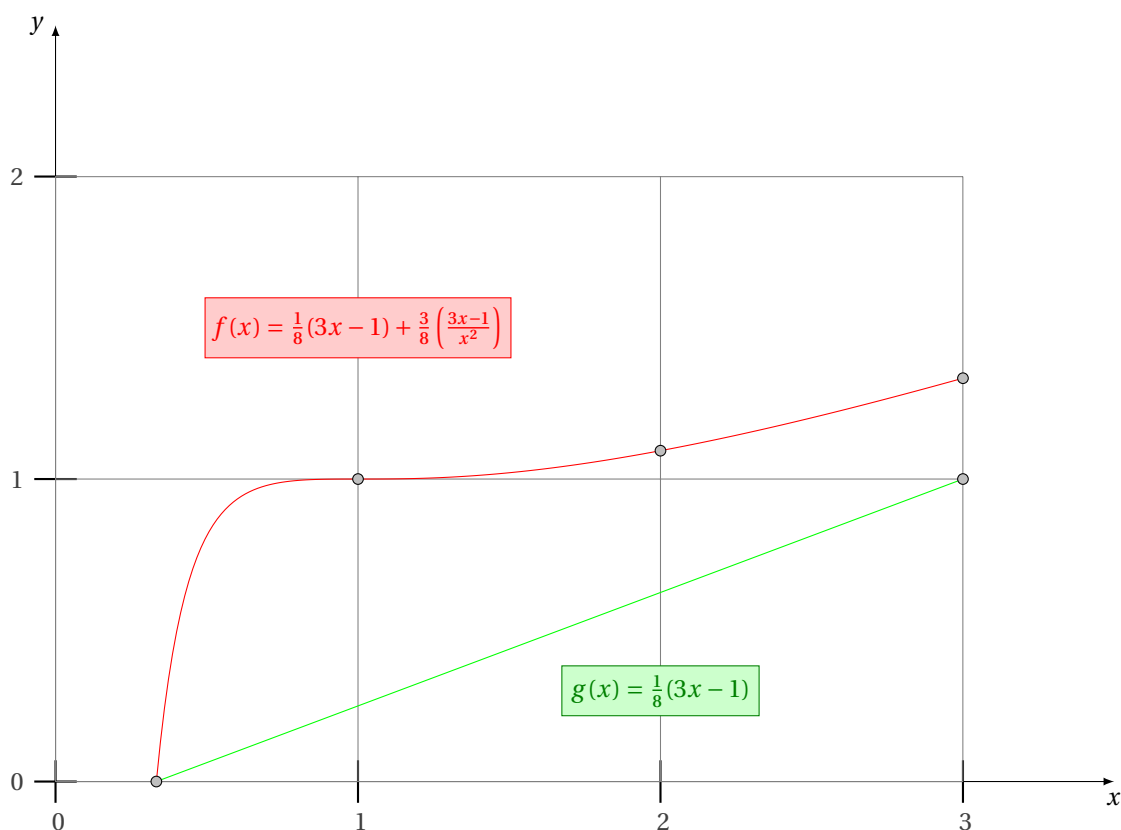


```
\begin{tikzpicture}[scale=1.6]
  \tkzInit[xmin = 0, xmax = 800,
    ymin = 0, ymax = 2000,
    xstep = 100, ystep = 400]
  \tkzGrid
  \tkzAxeXY
  \tkzFct[color = blue,
    domain = 0:800]%
    {(1./90000)*\x*\x*\x-(1./100)*\x*\x+(113./36)*\x}
  \foreach \va in {0,450,800}{%
    \tkzDefPointByFct[draw](\va)}
\end{tikzpicture}
```

## 5.6 设置点的样式

请参阅tkz-base.sty宏包的\tkzSetUpPoint命令和\tkzText命令。

```
\begin{tikzpicture}[scale=4]
\tkzInit[xmax=3,ymax=2]
\tkzAxeX
\tkzAxeY
\tkzGrid(0,0)(3,2)
\tkzFct[color = red,domain = 1./3:3]{0.125*(3*x-1)+0.375*(3*x-1)/(x*x)}
\tkzFct[color = green,domain = 1./3:3]{0.125*(3*x-1)}
\tkzSetUpPoint[shape=circle, size = 4, color=black, fill=lightgray]
\tkzDefPointByFct[draw,with = a](1)
\tkzDefPointByFct[draw,with = a](2)
\tkzDefPointByFct[draw,with = a](3)
\tkzDefPointByFct[draw,with = b](3)
\tkzDefPointByFct[draw,with = b](1/3)
\tkzText[draw,color= red,fill=red!20](1,1.5) %
    {\$f(x)=\frac{1}{8}(3x-1)+\frac{3}{8}\left(\frac{3x-1}{x^2}\right)%
    \left(\frac{3x-1}{x^2}\right)\$}
\tkzText[draw,color= green!50!black,fill=green!20]%
    (2,0.3){\$g(x)=\frac{1}{8}(3x-1)\$}
\end{tikzpicture}
```



## 6 标注

可以在绘制函数曲线后，为函数曲线添加标注，如函数  $f$  定义为：

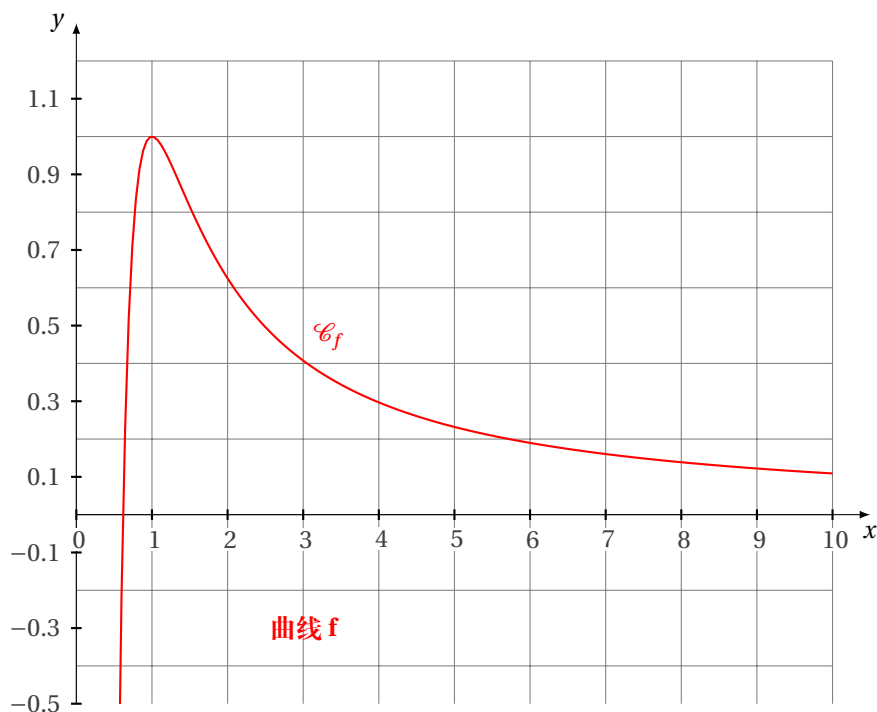
$$x > 0 \text{ et } f(x) = \frac{x^2 + 1}{x^3}$$

可以使用 **tkz-base** 宏包提供的 **\tkzText** 为函数添加标注，在添加标注时，可以为标注直接指定坐标。如果需要沿曲线添加标注，则最简单的方法是通过函数曲线定义一点，然后用该点为标注定位。

```
1 \tkzDefPointByFct(3)
2 \tkzText[above right](tkzPointResult){ $\mathcal{C}_f$ }
```

第 1 句代码使用函数定义了函数上的一个点，可以使用 **tkzPointResult** 得到该点。第 2 句代码使用定义的点为标注定位，当然，**TikZ** 能够对结果进行优化。

### 6.1 添加标注示例




```
\begin{tikzpicture}
  \tkzInit[xmin=0,xmax=10,
    ymin=-0.5,ymax=1.2,ystep=0.2]
  \tkzGrid
  \tkzAxeXY
  \tkzClip
  \tkzFct[thick,color=red,domain=0.55:10]{(\x*\x+\x-1)/(\x**3)}
  \tkzText(3,-0.3){\textbf{曲线}  $\mathbf{f}$ }
  \tkzDefPointByFct(3)
  \tkzText[above right,text=red](tkzPointResult){ $\mathcal{C}_f$ }
\end{tikzpicture}
```



## 7 切线绘制命令

对第1个函数命名为`\tkzFcta`，第2个函数命名`\tkzFctb`，以此类推…。如果一个环境中有多函数，则可以使用`with`指定需要使用的函数。

 注意在使用`\tkzFct`命令和`\tkzDrawTangentLine`命令之前，需要使用`\tkzInit`命令初始化绘图环境。可以用`kl`和`kr`系数指定半切线矢量长度，如`kl=0`或`kr=0`表示取消对应的半切线(`l=left`, `r=right`)。如果`xstep=1`和`ystep=1`，斜率为1，则半切线的长度是 $\sqrt{2}$ 。其他情况下，如果半切线向量 $\vec{AT}$ 的斜率是 $p$ ，则半切线的坐标是 $(kl, kl \cdot p)$ 。

7.1 `\tkzDrawTangentLine`命令：绘制函数切线

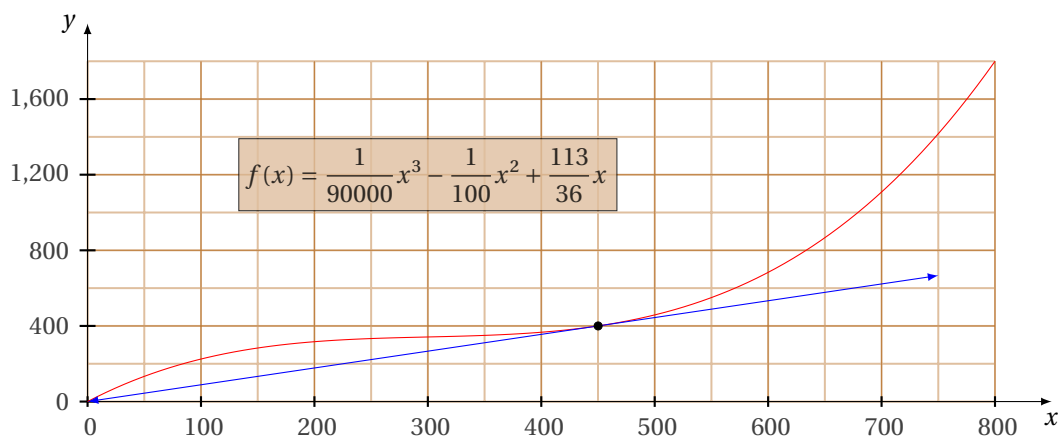
`\tkzDrawTangentLine[< 命令选项>](<a>)`

如果直接在`\tkzFct`命令后使用该命令，则表示使用当前函数。否则，需要用`with`指定函数。

参数	样例	说明
a	<code>\tkzDrawTangentLine(0)</code>	横坐标为 0 的函数点处的切线

可以使用所有类似`color`或`style`的有效TikZ选项。

选项	默认值	含义
draw	false	布尔值，如为 true，则绘制切点
with	a	选择指定的函数
kr	1	右半切线长度系数
kl	1	左半切线长度系数

7.2 `xstep`和`ystep`不为 1 的切线

要说明的是，没有必要进行计算坐标真实值，只需要直接指定与刻度对应的坐标值。

可以使用如下类似代码修改切线的样式：

`\tikzset{tan style/.style={-}}` 的默认设置为：

`\tikzset{tan style/.style={->,>=latex}}`

```
\begin{tikzpicture}[xscale=1.5]
\tikzset{tan style/.style={-}}
\tkzInit[xmin=0,xmax=800,xstep=100,
```

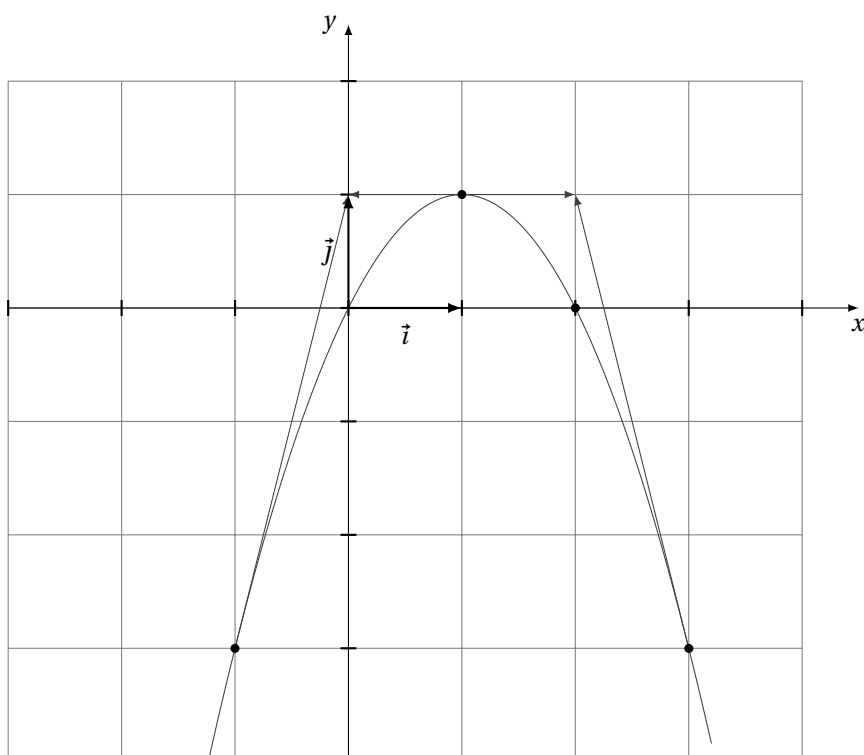
```

ymin=0,ymax=1800,ystep=400]
\tkzGrid[color=brown,sub,subxstep=50,subystep=200](0,0)(800,1800)
\tkzAxeXY
\tkzFct[color=red,samples=100,domain = 0:800]%
  {(1./90000)*\x*\x*\x-(1./100)*\x*\x+(113./36)*\x}
\tkzDrawTangentLine[color=blue,kr=300,kl=450,coord](450)
\tkzText[draw,color = black,%
  fill = brown!50,opacity = 0.8](300,1200)%
{$f(x)=\dfrac{1}{90000}x^3-\dfrac{1}{100}x^2+\dfrac{113}{36}x$}
\end{tikzpicture}

```

### 7.3 kl、kr和draw选项

如果kl或kr为0，则只绘制半切线，否则，可以用数字表示切线长度与原始切线长度的百分比。draw选项表示需要绘制切点。



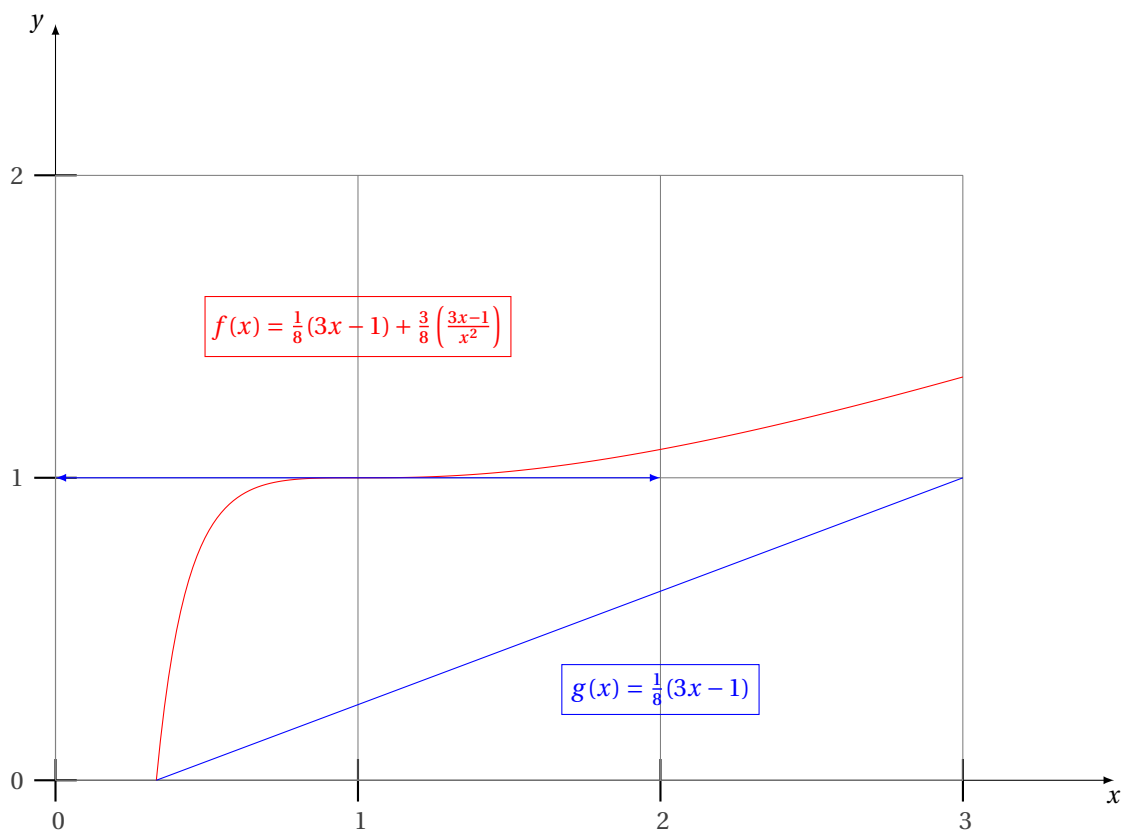
```

\begin{tikzpicture}[scale=1.5]
\tkzInit[xmin=-3,xmax=4,ymin=-4,ymax=2]
\tkzGrid \tkzDrawXY \tkzClip
\tkzFct[domain = -2.15:3.2]{(-x*x)+2*x}
\tkzDefPointByFct[draw](2)
\tkzDrawTangentLine[kl=0,draw](-1)
\tkzDrawTangentLine[draw](1)
\tkzDrawTangentLine[kr=0,draw](3)
\tkzRep
\end{tikzpicture}

```

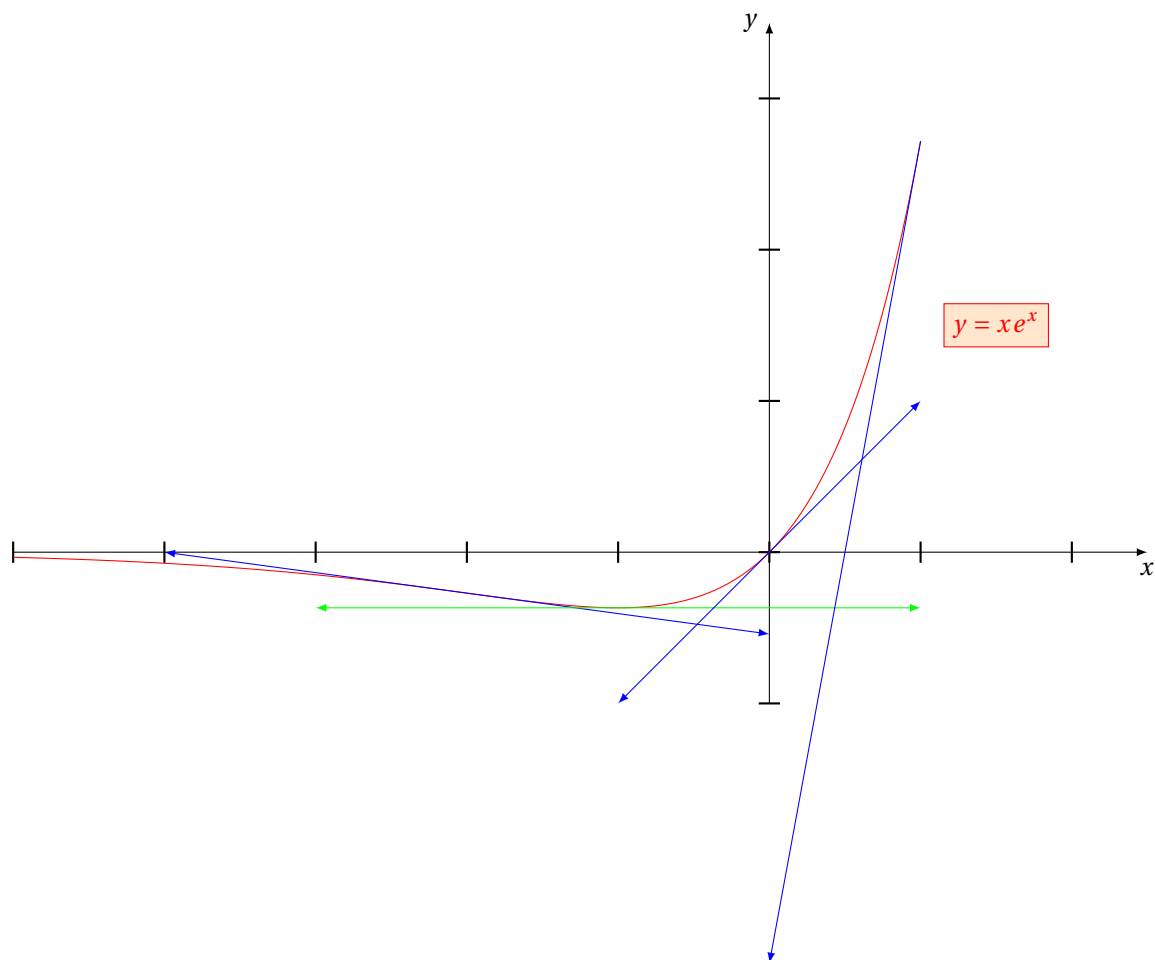
## 7.4 with选项

可以将`\tkzDrawTangentLine`放在第一个函数定义代码行之后,表示使用该函数(*a*),当然,也可以使用`with`选项指定需要绘制切线的函数。



```
\begin{tikzpicture}[scale=4]
\tkzInit[xmax=3,ymax=2]
\tkzAxeXY
\tkzGrid(0,0)(3,2)
\tkzFct[color = red, domain = 1/3:3]{0.125*(3*x-1)+0.375*(3*x-1)/(x*x)}
\tkzFct[color = blue, domain = 1/3:3]{0.125*(3*x-1)}
\tkzDrawTangentLine[with=a,
                    color=blue](1)
\tkzText[draw,
        color= red](1,1.5)%
        {$f(x)=\frac{1}{8}(3x-1)+\frac{3}{8}\left(\frac{3x-1}{x^2}\right)$}
\tkzText[draw,
        color= blue](2,0.3)%
        {$g(x)=\frac{1}{8}(3x-1)$}
\end{tikzpicture}
```

## 7.5 切线示例



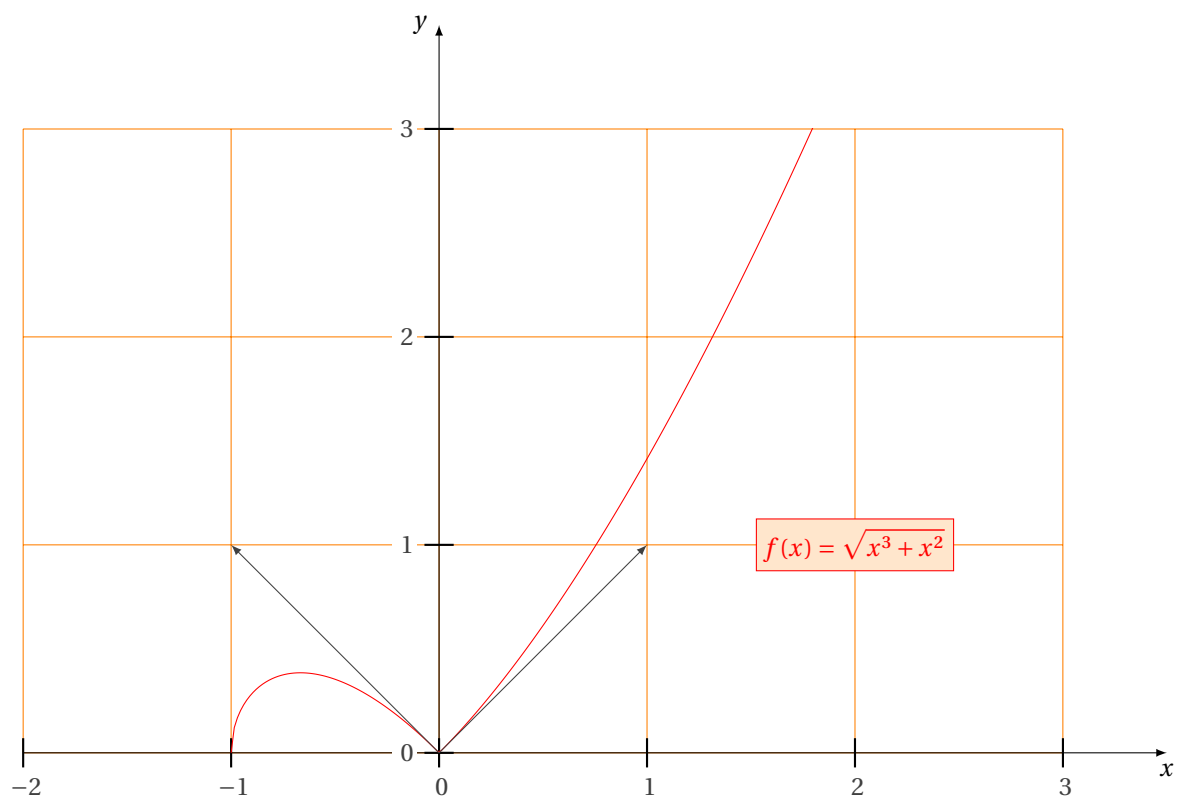
```
\begin{tikzpicture}[scale=2]
  \tkzInit[xmin=-5,xmax=2,ymin=-1,ymax=3]
  \tkzDrawX
  \tkzDrawY
  \tkzText[draw,color = red,fill = orange!20]( 1.5,1.5){$y = xe^x$}
  \tkzFct[color = red, domain = -5:1]{x*exp(x)}%
  \tkzDrawTangentLine[color=blue,kr=2,kl=2](-2)
  \tkzDrawTangentLine[color=green,kr=2,kl=2](-1)
  \tkzDrawTangentLine[color=blue](0)
  \tkzDrawTangentLine[color=blue,kr=0](1)
\end{tikzpicture}
```

## 7.6 半切线

注意，切线实际上有两条半切线，例如对于函数：

$$(((x+1)*x)*x)**0.5((x**3+x**2)**0.5或(x*x*x+x*x)**(0.5)).$$

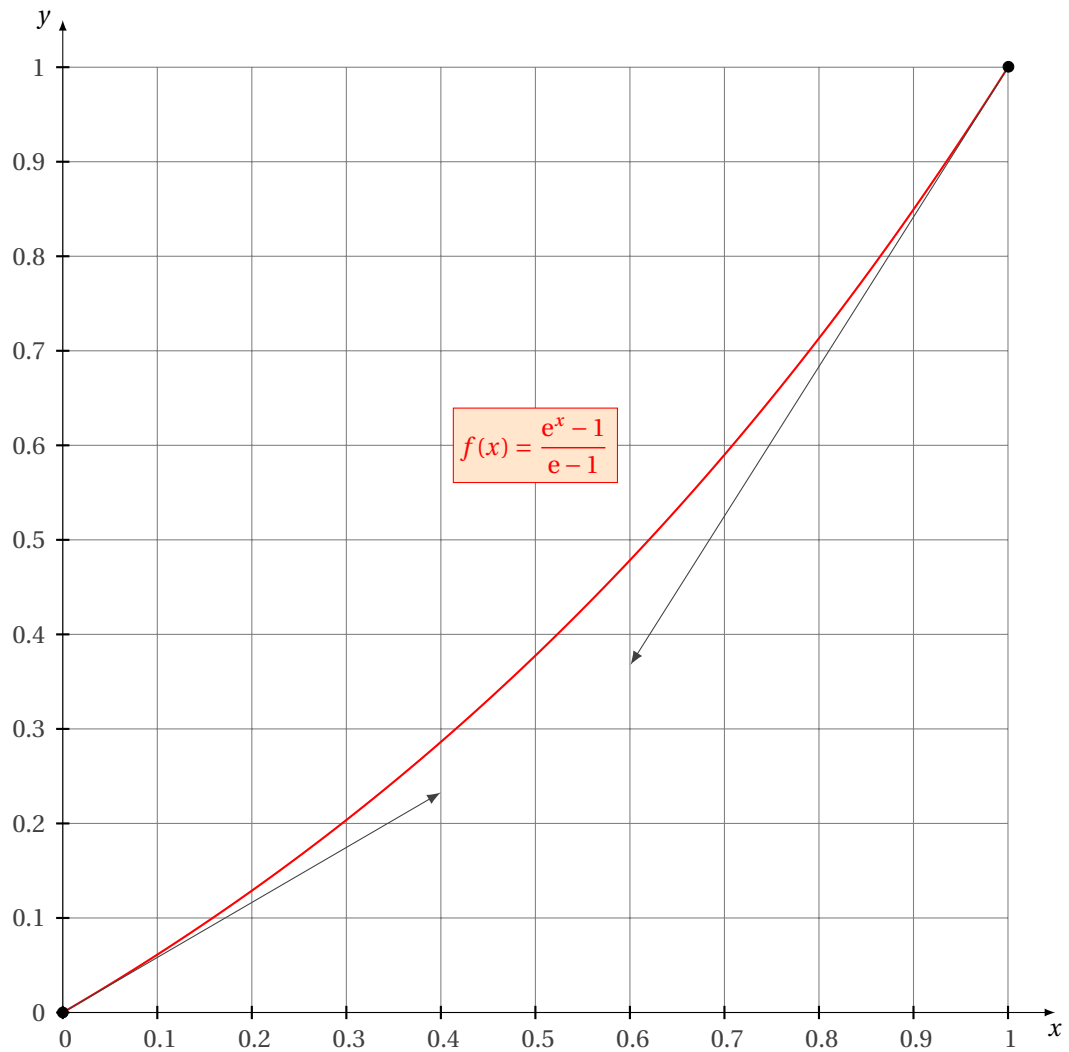
在本例中，能够自动得到两条半切线。



```
\begin{tikzpicture}[scale=2.75]
  \tkzInit[xmin=-2,xmax=3,ymax=3]
  \tkzGrid[color=orange](-2,0)(3,3)
  \tkzAxeX
  \tkzAxeY
  \tkzFct[color = red ,domain = -1:2]{(((x+1)*x)*x)**0.5}
  \tkzDrawTangentLine(0)
  \tkzText[draw,color = red,fill = orange!20](2,1){$f(x)=\sqrt{x^3+x^2}$}
\end{tikzpicture}
```

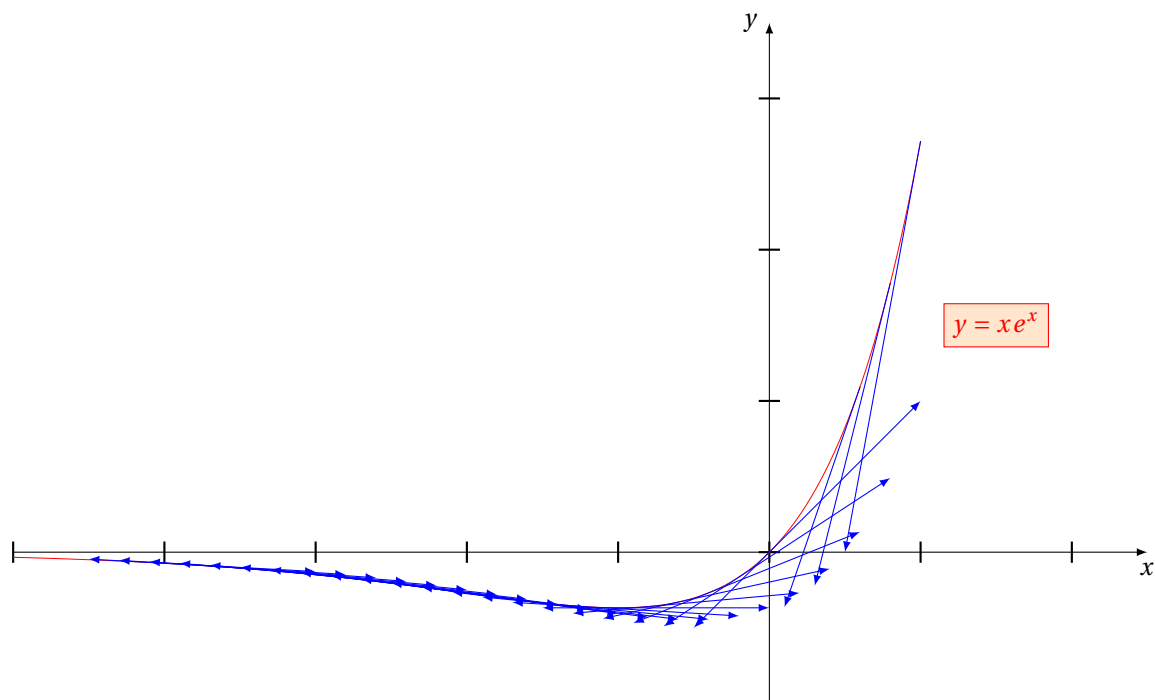
### 7.7 Lorentz 曲线的半切线

在此，只需得到 0 和 1 之间的半切线，仅需为 **kr** 和 **kl** 赋合适的值即可。



```
\begin{tikzpicture}[scale=1.25]
  \tkzInit[xmax=1,ymax=1,xstep=0.1,ystep=0.1]
  \tkzGrid(0,0)(1,1)
  \tkzAxeXY
  \tkzFct[color = red,thick, domain =0:1]{(exp(\x)-1)/(exp(1)-1)}
  \tkzSetUpPoint[size=4]
  \tkzDrawTangentLine[draw, kl = 0, kr = 0.4](0)
  \tkzDrawTangentLine[draw, kl = 0.4,kr = 0 ](1)
  \tkzText[draw,color = red,fill = orange!20](0.5,0.6)%
    {\$f(x)=\dfrac{\text{e}^x-1}{\text{e}-1}\$}
\end{tikzpicture}
```

## 7.8 切线族示例



```
\begin{tikzpicture}[scale=2]
  \tikzstyle{tan style}=[-]
  \tkzInit[xmin=-5,xmax=2,ymin=-1,ymax=3]
  \tkzDrawXY
  \tkzText[draw,color = red, fill = orange!20](1.5,1.5){$y = xe^x$}
  \tkzFct[line width = 0.01 pt,color = red, domain = -5:1]{x*exp(x)}
  \foreach \x in {-4,-3.8,...,0}{%
    \tkzDrawTangentLine[color=blue,line width=.4pt,kr=1,kl=0.5](\x)}
  \foreach \x in {0.6,0.8,1}{%
    \tkzDrawTangentLine[color=blue,line width=.4pt, kr=0,kl=0.5](\x)}
\end{tikzpicture}
```

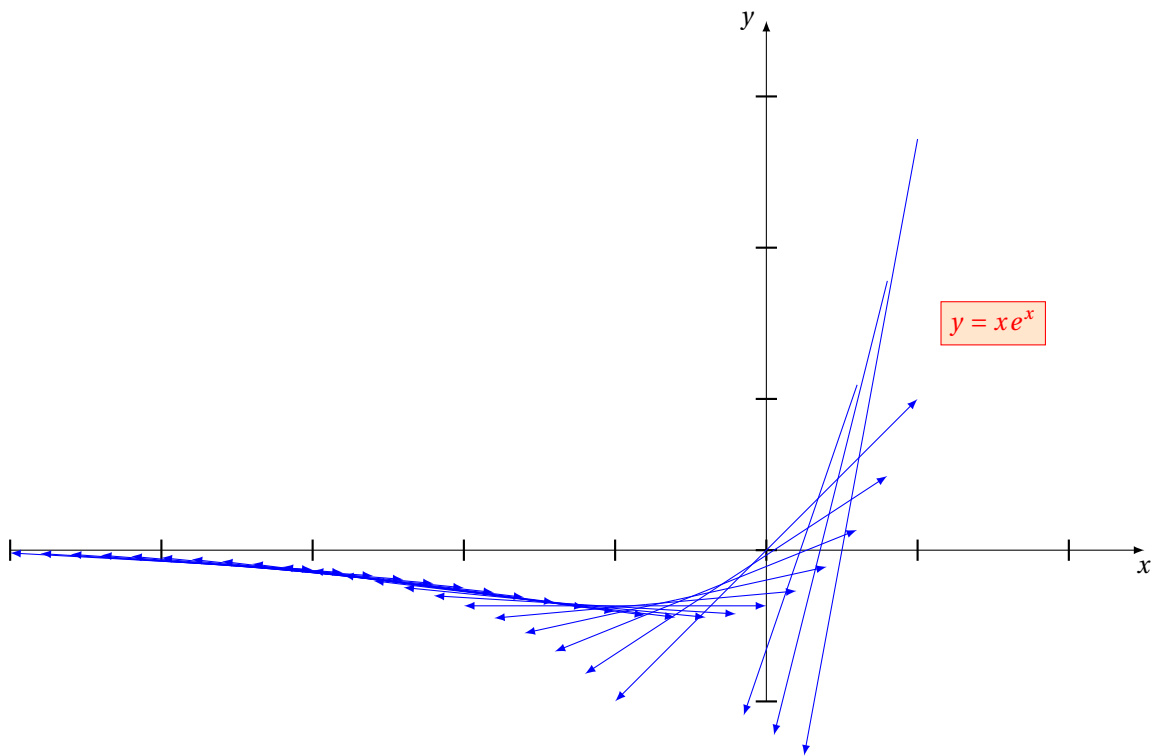
## 7.9 无曲线的切线族

为此，必须使用符合`fp.sty`宏包语法的表达式定义函数。

需要定义`\tkzFctLast`全局宏。

```
\global\edef\tkzFctLast{x*exp(x)}
```

## 7.9.1 \tkzFctLast宏



```

\begin{tikzpicture}[scale=2]
\tikzstyle{tan style}=[-]
\tkzInit[xmin=-5,xmax=2,ymin=-1,ymax=3]
\tkzDrawXY
\tkzText[draw,color = red, fill = orange!20](1.5,1.5){$y = xe^x$}
\global\edef\tkzFctLast{x*exp(x)}% 这一行代码非常重要
\foreach \v in {-4,-3.8,...,0}{%
\tkzDrawTangentLine[color=blue,line width=.4pt,kl=1](\v)}
\foreach \v in {0.6,0.8,1}{%
\tkzDrawTangentLine[color=blue,line width=.4pt,kr=0,kl=.75](\v)}
\end{tikzpicture}

```



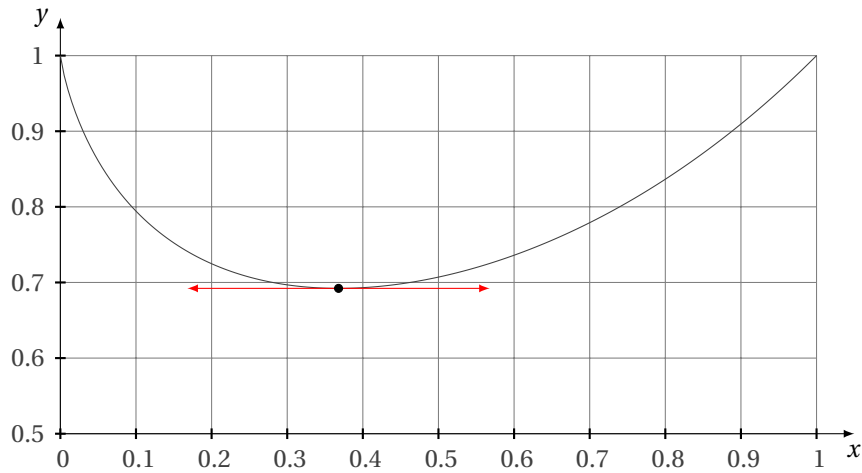
### 7.10 计算的历史记录

如果在参数中使用带有括号的表达式，则会出现问题。如 $\{1/\exp(1)\}$ 是正确的，但 $(1/\exp(1))$ 则会引发错误。在后续计算中，也需要注意该问题。

#### 7.10.1 提前定义计算结果

```
\FPeval\vx{1/exp(1)}
```

#### 7.10.2 使用计算结果



```
\begin{tikzpicture}[scale=1]
  \tkzInit[xmax=1,xstep=0.1,ymin=0.5,ymax=1,ystep=0.1]
  \tkzGrid      \tkzAxeXY
  \tkzFct[domain = 0.00001:1]{(\x**\x)}
  \tkzDrawTangentLine[draw,color = red, kr = 0.2,kl = 0.2]({1/exp(1)})
\end{tikzpicture}
```

8 区域填充命令

例如，可能通过着色表示一个函数曲线与坐标横轴、直线  $x = a$  和  $x = b$  构成的区域。

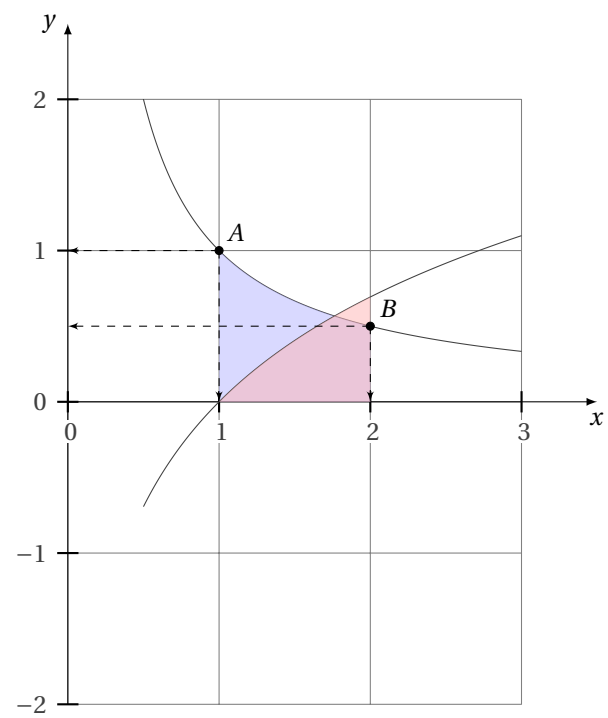
8.1 \tkzDrawArea或\tkzArea命令：区域填充

\tkzDrawArea[< 命令选项>]

可以使用所有有效 TikZ 选项。

选项	默认值	含义
domain	-5:5	函数定义域
with	a	指定函数
color	200	颜色
opacity	无	透明度
style	black	线型

8.2 对数函数

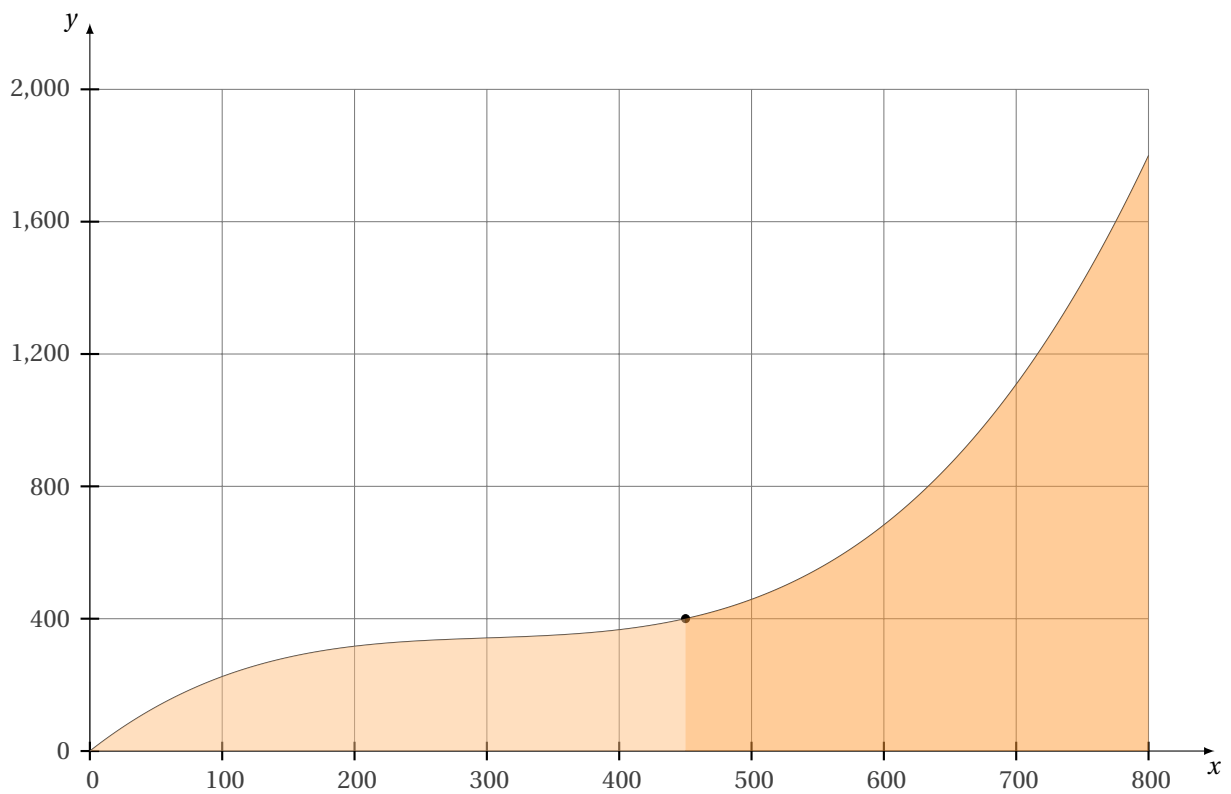


```

\begin{tikzpicture}[scale=2]
  \tkzInit[xmin=0,xmax=3,xstep=1,
           ymin=-2,ymax=2,ystep=1]
  \tkzGrid
  \tkzAxeXY
  \tkzFct[domain= 0.4:3]{1./x}
  \tkzDefPointByFct(1)
  \tkzGetPoint{A}
  \tkzDefPointByFct(2)
  \tkzGetPoint{B}
  \tkzLabelPoints[above right](A,B)
  \tkzDrawArea[color=blue!30,
               domain = 1:2]
  \tkzFct[domain = 0.5:3]{log(x)}
  \tkzDrawArea[color=red!30,
               domain = 1:2]
  \tkzPointShowCoord(A)
  \tkzPointShowCoord(B)
  \tkzDrawPoints(A,B)
\end{tikzpicture}

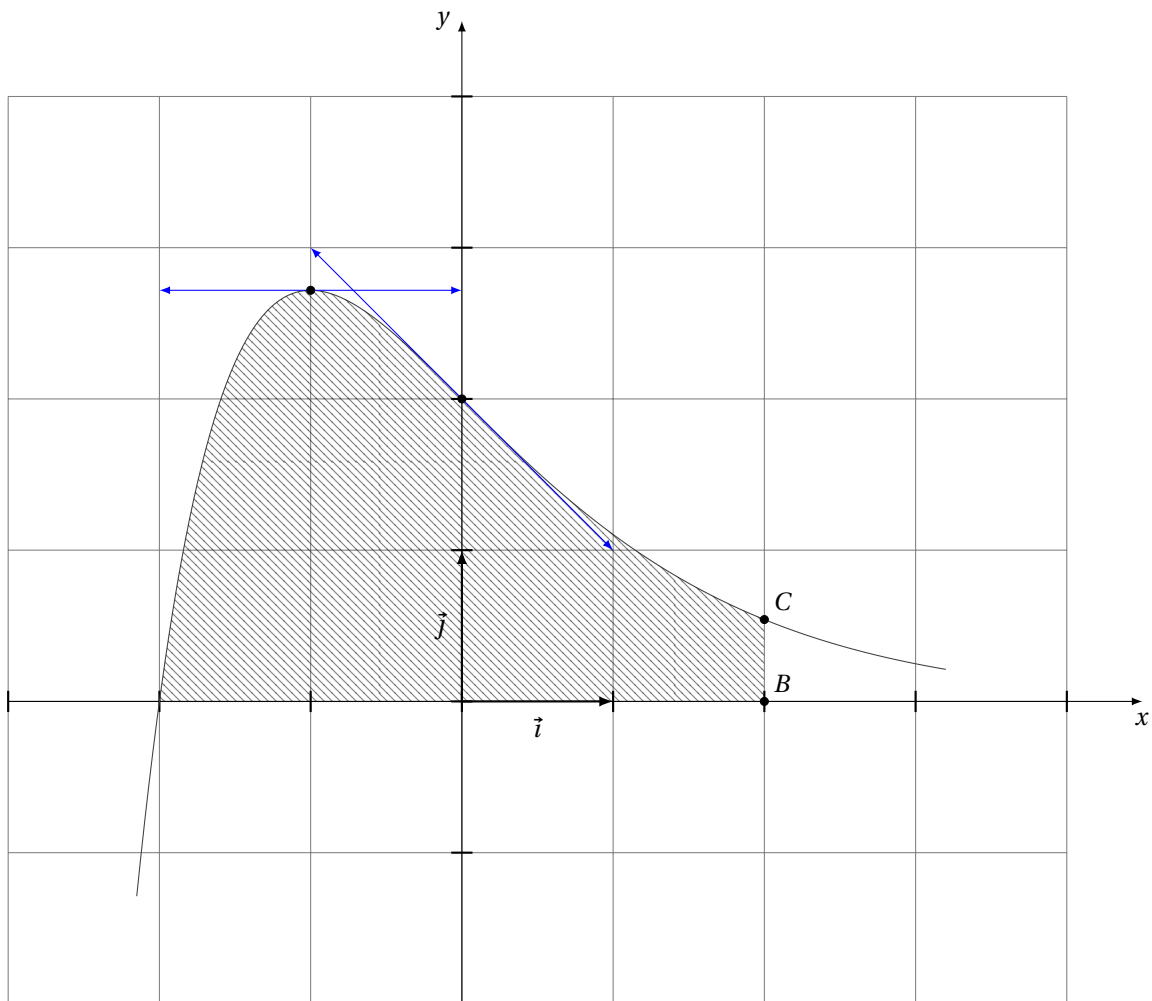
```

### 8.3 简单示例



```
\begin{tikzpicture}[scale=1.75]
\tkzInit[xmin=0,xmax=800,xstep=100,
        ymin=0,ymax=2000,ystep=400]
\tkzGrid
\tkzAxeXY
\tkzFct[domain = 0:800]{(1./90000)*\x*\x*\x-(1./100)*\x*\x+(113./36)*\x}
\tkzDefPoint(450,400){a}
\tkzDrawPoint(a)
\tkzDrawArea[color=orange!50, domain =0:450]
\tkzDrawArea[color=orange!80, domain =450:800]
\end{tikzpicture}
```

## 8.4 填充样式



```

\begin{tikzpicture}[scale=2]
  \tkzInit[xmin=-3,xmax=4,ymin=-2,ymax=4]
  \tkzGrid(-3,-2)(4,4)
  \tkzDrawXY
  \tkzFct[domain = -2.15:3.2]{(2+\x)*exp(-\x)}
  \tkzDrawArea[pattern=north west lines,domain =-2:2]
  \tkzDrawTangentLine[draw,color=blue](0)
  \tkzDrawTangentLine[draw,color=blue](-1)
  \tkzDefPointByFct(2) \tkzGetPoint{C}
  \tkzDefPoint(2,0){B}
  \tkzDrawPoints(B,C) \tkzLabelPoints[above right](B,C)
  \tkzRep
\end{tikzpicture}

```

8.5 `\tkzDrawAreafg`命令：填充两条函数曲线之间的区域`\tkzDrawAreafg`[< 命令选项>]

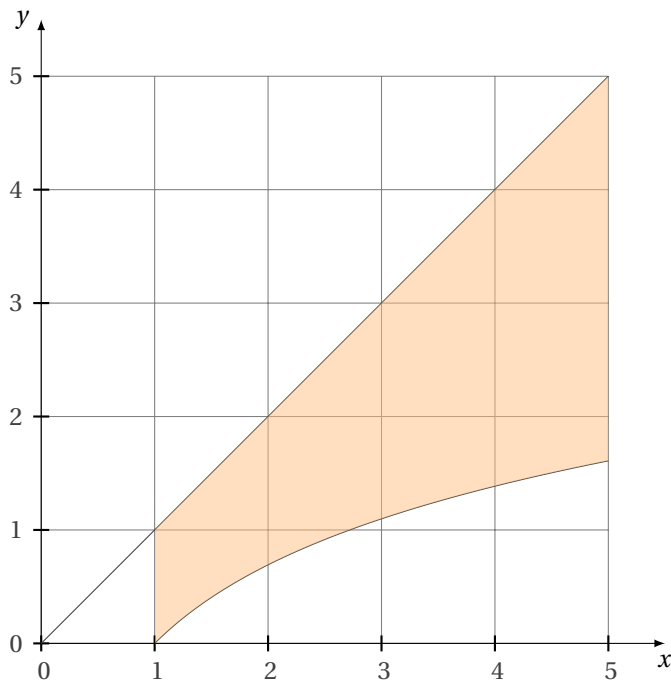
该命令用于填充两条函数曲线包围的区域，要求曲线 (a) 必须高于曲线 (b)。

选项	默认值	说明
<code>between</code>	<code>a and b</code>	曲线名称
<code>domain= min:max</code>	<code>domain=-5:5</code>	定义域，是 TikZ 的选项
<code>opacity</code>	<code>0.5</code>	透明度

可以使用所有有效的 TikZ 填充样式 `pattern` 选项。

## 8.6 两条函数曲线之间的区域

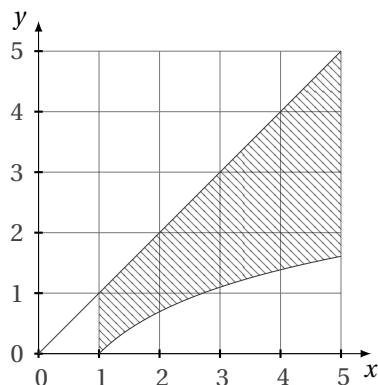
默认情况下，区域位于两条函数曲线之间。



```
\begin{tikzpicture}[scale=1.5]
  \tkzInit[xmax=5,ymax=5]
  \tkzGrid \tkzAxeXY
  \tkzFct[domain = 0:5]{x}
  \tkzFct[domain = 1:5]{log(x)}
  \tkzDrawAreafg[color = orange!50,domain = 1:5]
\end{tikzpicture}
```

## 8.7 设置填充图案

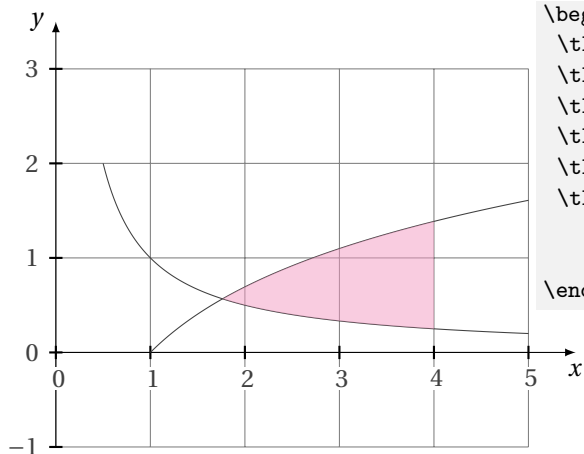
```
\tkzDrawAreafg[between= a and b,pattern=north west lines,domain = 1:5]
```



```
\begin{tikzpicture}[scale=.8]
  \tkzInit[xmax=5,ymax=5]
  \tkzGrid
  \tkzAxeXY
  \tkzFct[domain = 0:5]{x}
  \tkzFct[domain = 1:5]{log(x)}
  \tkzDrawAreafg[between= a and b,pattern=north west lines,domain = 1:5]
\end{tikzpicture}
```

## 8.8 between选项

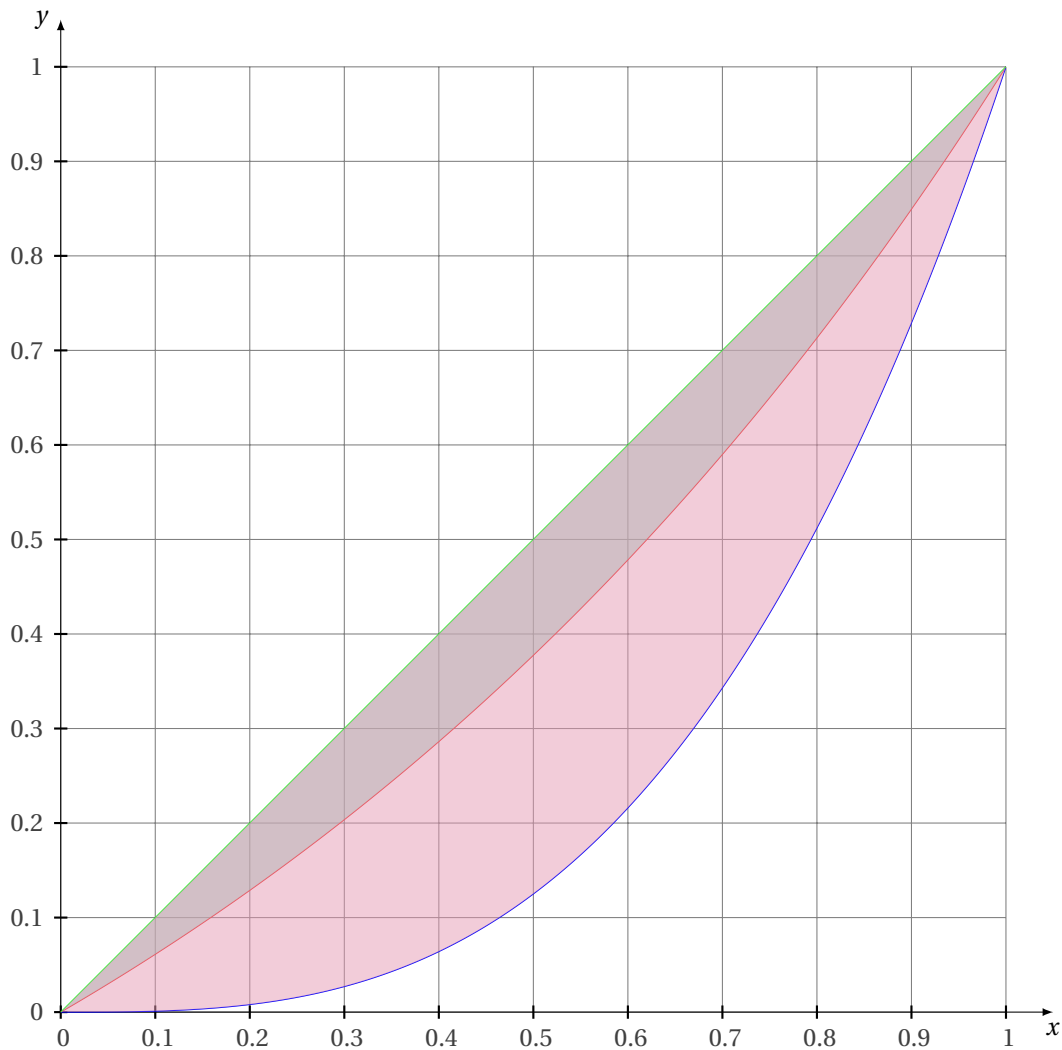
注意，**between**的引用顺序，在此表示函数曲线 (b) 高于函数曲线 (a)。



```
\begin{tikzpicture}[scale=1.25]
  \tkzInit[ymin=-1,xmax=5,ymax=3]
  \tkzGrid
  \tkzAxeXY
  \tkzFct[domain = 0.5:5]{1/x}% courbe a
  \tkzFct[domain = 1:5]{log(x)}% courbe b
  \tkzDrawAreafg[between=b and a,
    color=magenta!50,
    domain = 1:4]
\end{tikzpicture}
```

## 8.9 两条 Lorentz 函数曲线之间的区域

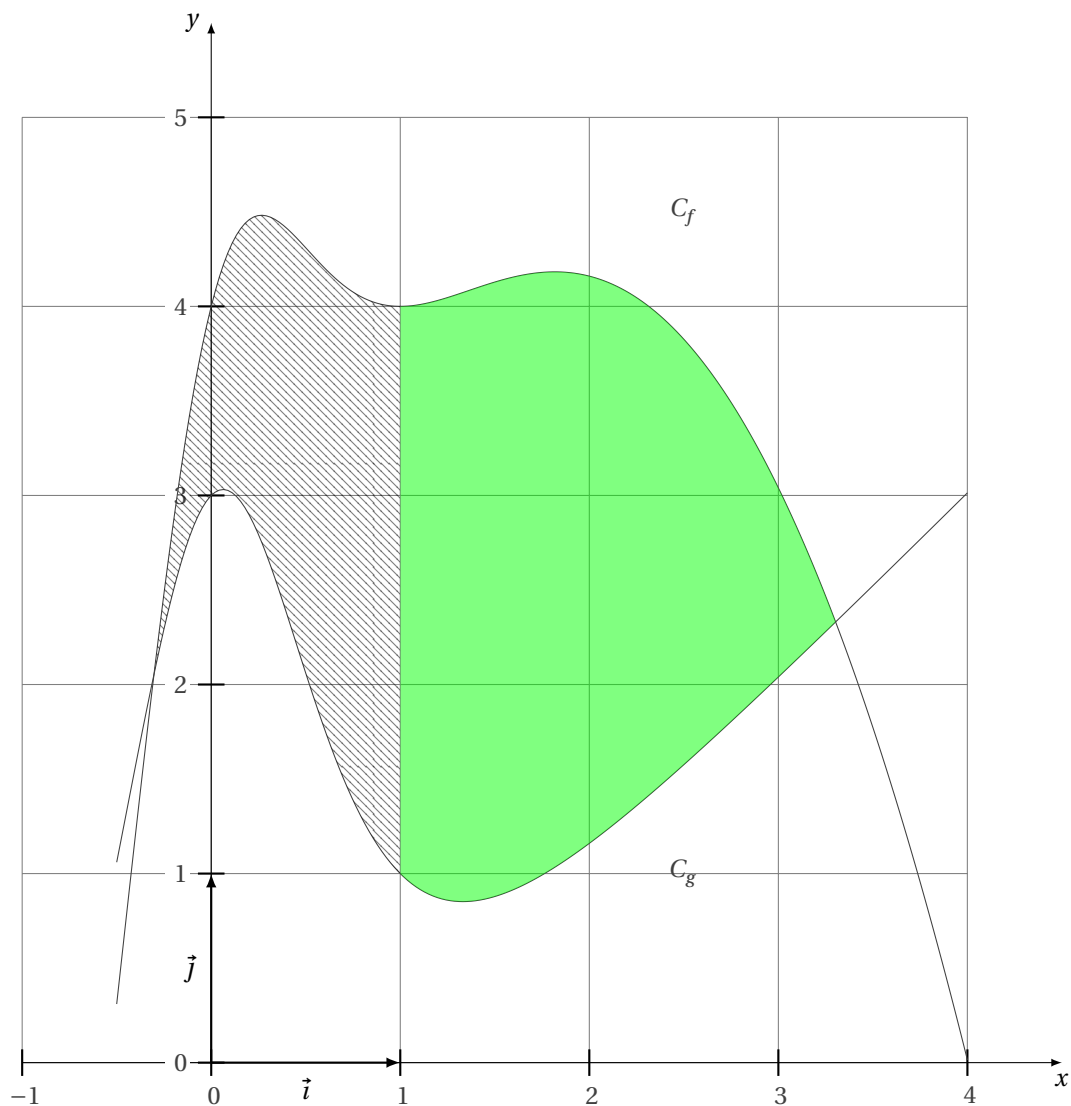
在此,也要注意**between**选项中的函数曲线引用顺序。



```
\begin{tikzpicture}[scale=1.25]
  \tkzInit[xmax=1,ymax=1,xstep=0.1,ystep=0.1]
  \tkzGrid
  \tkzAxeXY
  \tkzFct[color = red,domain = 0:1]{(exp(\x)-1)/(exp(1)-1)}
  \tkzFct[color = blue,domain = 0:1]{\x*\x*\x}
  \tkzFct[color = green,domain = 0:1]{\x}
  \tkzDrawAreafg[between = c and b,color=purple!40,domain = 0:1]
  \tkzDrawAreafg[between = c and a,color=gray!60,domain = 0:1]
\end{tikzpicture}
```



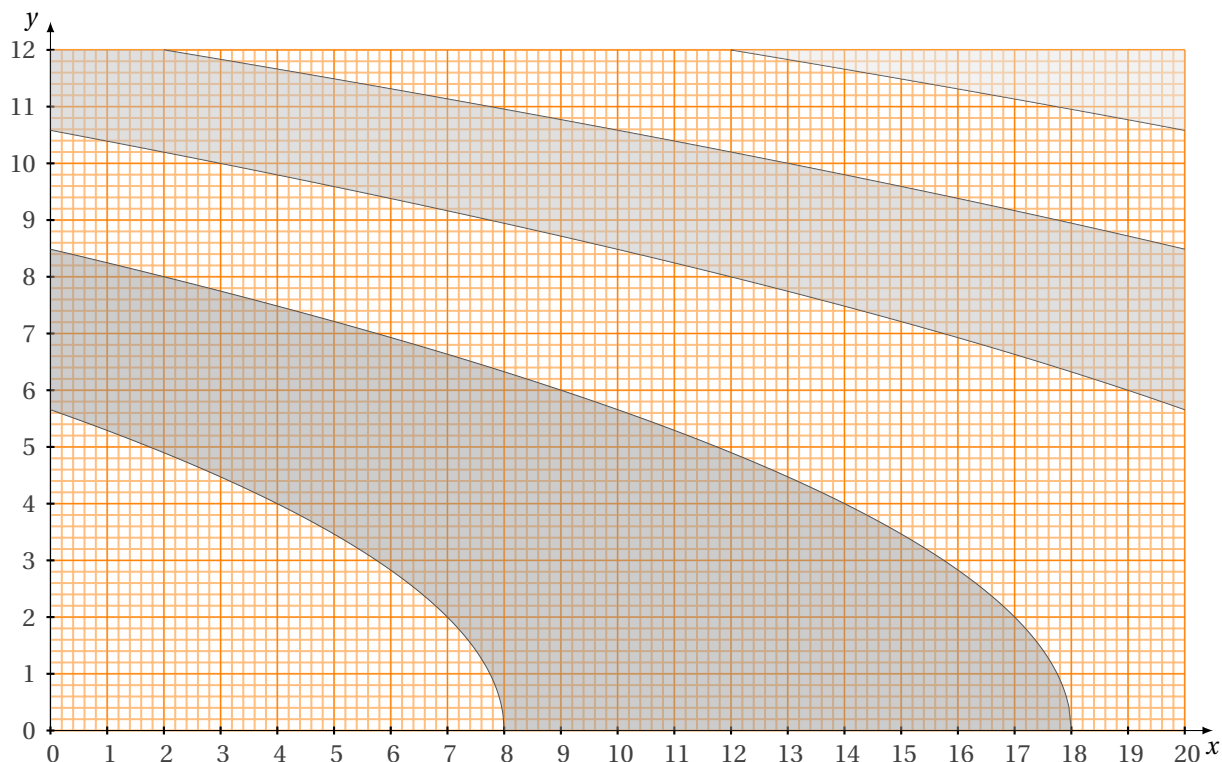
## 8.10 混合样式



```
\begin{tikzpicture}[scale=2.5]
  \tkzInit[xmin=-1,xmax=4,ymin=0,ymax=5]
  \tkzGrid
  \tkzAxeXY
  \tkzFct[domain = -.5:4]{ 4*x-x**2+4/(x**2+1)**2}
  \tkzFct[domain = -.5:4]{x-1+4/(x**2+1)**2}
  \tkzDrawAreafg[color=green,domain = 1:4]
  \tkzDrawAreafg[pattern=north west lines,domain = -.5:1]
  \tkzRep
  \tkzText(2.5,4.5){$C_f$}
  \tkzText(2.5,1){$C_g$}
\end{tikzpicture}%
```

## 8.11 水平曲线

注意代码第 10 行和第 11 行的常量函数定义。



```

1 \begin{tikzpicture}[scale=.75]
2   \tkzInit[xmax=20,ymax=12]
3   \tkzGrid[color=orange,sub](0,0)(20,12)
4   \tkzAxeXY
5   \tkzFct[samples=400,domain =0:8]{(32-4*x)**(0.5)} % a
6   \tkzFct[samples=400,domain =0:18]{(72-4*x)**(0.5)} % b
7   \tkzFct[samples=400,domain =0:20]{(112-4*x)**(0.5)} % c
8   \tkzFct[samples=400,domain =2:20]{(152-4*x)**(0.5)} % d
9   \tkzFct[samples=400,domain =12:20]{(192-4*x)**(0.5)} % e
10  \def\tkzFctgnuf{0} % f
11  \def\tkzFctgnug{12} % g
12  \tkzDrawAreafg[between= b and a,color=gray!80,domain = 0:8]
13  \tkzDrawAreafg[between= b and f,color=gray!80,domain = 8:18]
14  \tkzDrawAreafg[between= d and c,color=gray!50,domain = 2:20]
15  \tkzDrawAreafg[between= g and c,color=gray!50,domain = 0:2]
16  \tkzDrawAreafg[between= g and e,color=gray!20,domain =12:20]
17 \end{tikzpicture}%

```

## 9 黎曼和

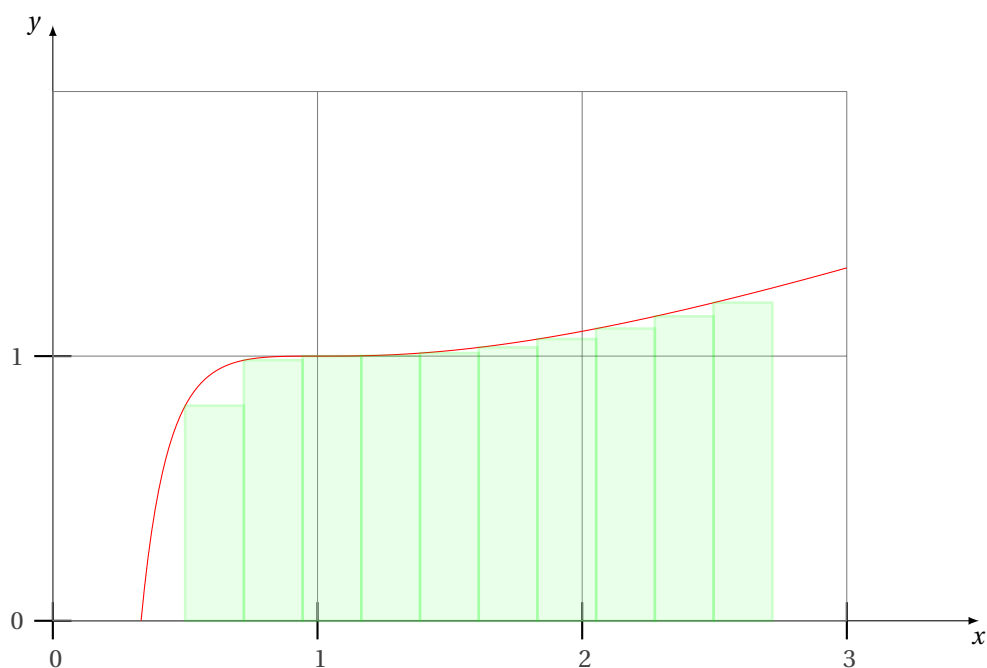
`\tkzDrawRiemannSum[< 命令选项>]{f(t)}`

该命令用于绘制函数的黎曼和矩形，可以使用所有有效 TikZ 选项。

选项	默认值	含义
<code>interval</code>	<code>1:2</code>	间隔区间
<code>number</code>	<code>10</code>	了间隔数

可以将四个命令组合在一起，并使用相同的选项。

## 9.1 求黎曼和

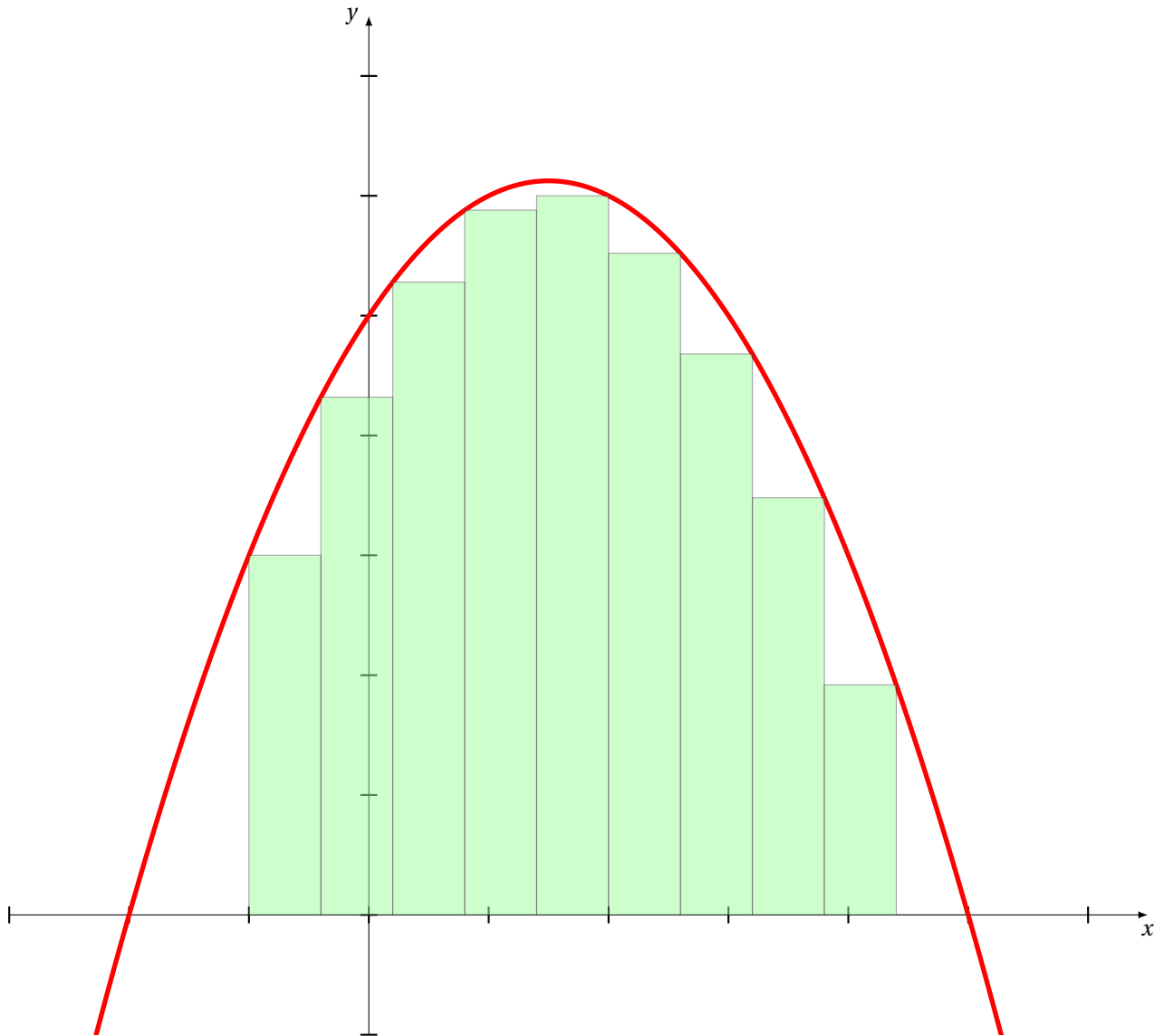


```
\begin{tikzpicture}[scale=3.5]
\tkzInit[xmax=3,ymax=1.75]
\tkzAxeXY
\tkzGrid(0,0)(3,2)
\tkzFct[color = red, domain =1/3:3]{0.125*(3*x-1)+0.375*(3*x-1)/(x*x)}
\tkzDrawRiemannSum[fill=green!40,opacity=.2,color=green,
                    line width=1pt,interval=1./2:exp(1),number=10]
\end{tikzpicture}
```

`\tkzDrawRiemannSumInf` [`< 命令选项>`]

是前一个命令变体，但矩形始终位于函数曲线下方。

## 9.2 `\tkzDrawRiemannSumInf` 命令示例

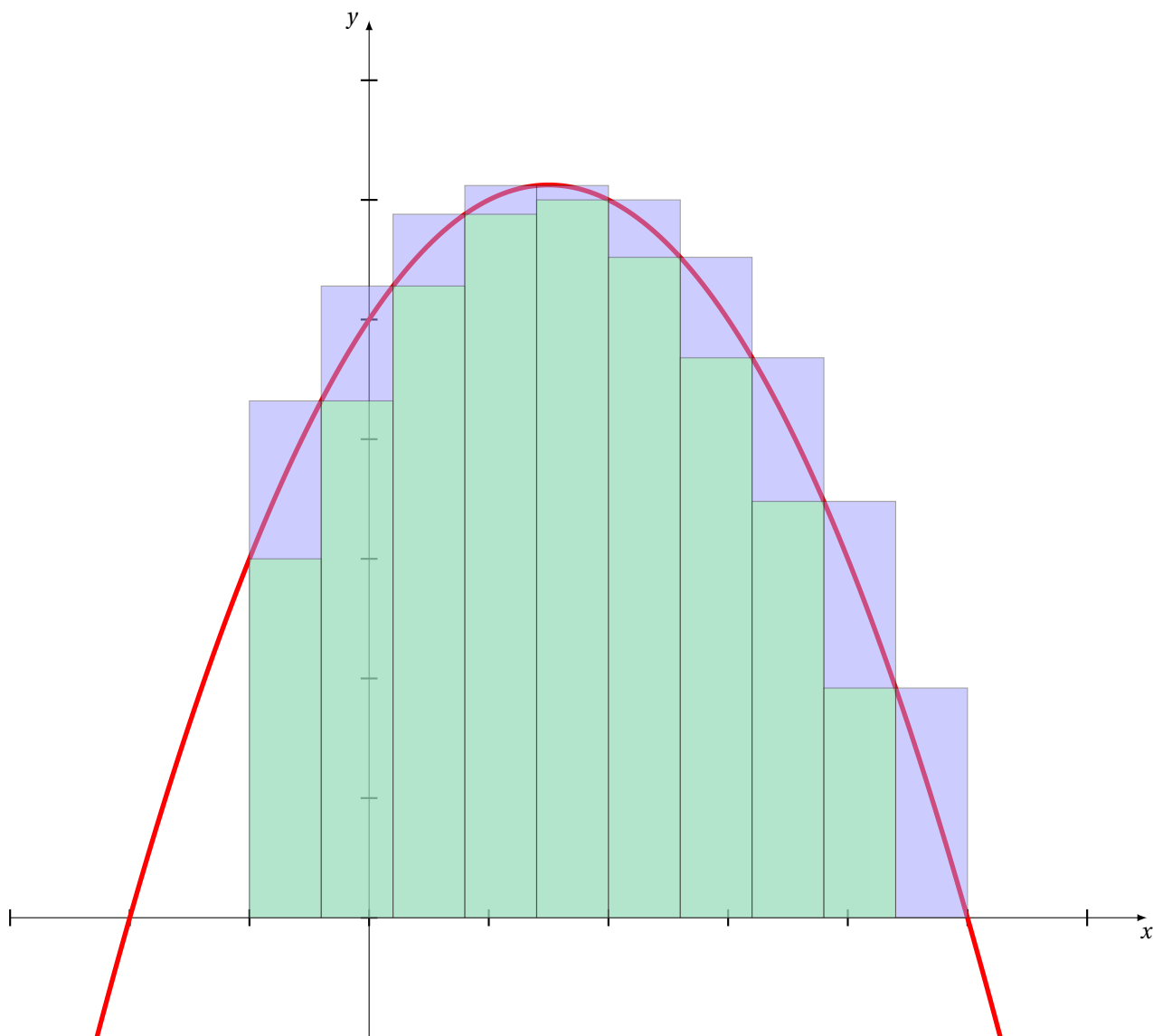


```
\begin{tikzpicture}[scale=1.75]
\tkzInit[xmin=-3,xmax=6,ymin=-2,ymax=14,ystep=2]
\tkzDrawX \tkzDrawY
\tkzFct[line width=2pt,color = red, domain =-3:6]{(-\x-2)*(\x-5)}
\tkzDrawRiemannSumInf[fill=green!40,opacity=.5,interval=-1:5,number=10]
\end{tikzpicture}
```

`\tkzDrawRiemannSumSup`[< 命令选项>]

第 1 个命令的变体，但矩形始终位于函数曲线上方。

### 9.3 `\tkzDrawRiemannSumSup`命令示例

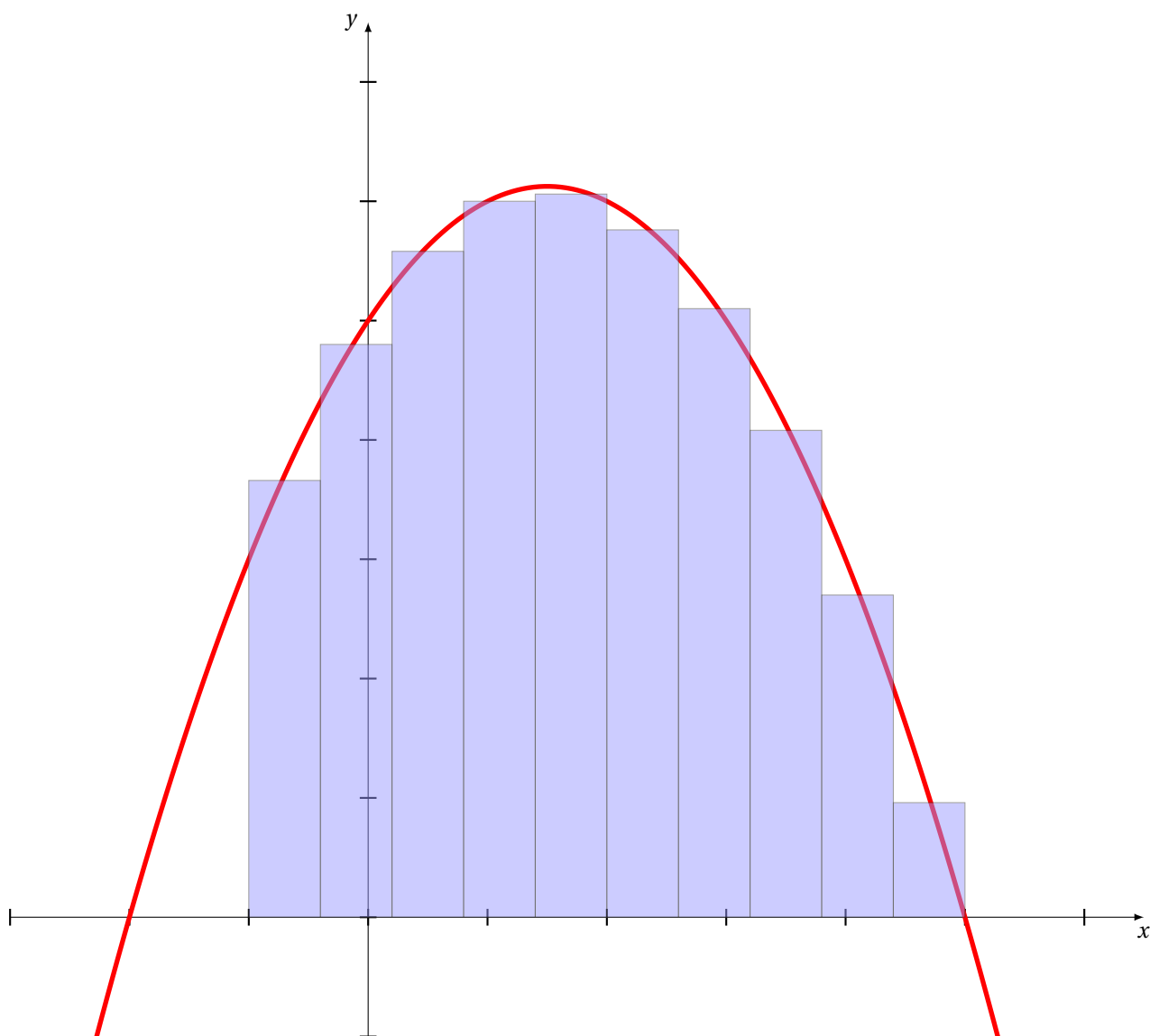


```
\begin{tikzpicture}[scale=1.75]
  \tkzInit[xmin=-3,xmax=6,ymin=-2,ymax=14,ystep=2]
  \tkzDrawX \tkzDrawY
  \tkzFct[line width=2pt,color = red, domain =-3:6]{(-\x-2)*(\x-5)}
  \tkzDrawRiemannSumSup[fill=blue!40,opacity=.5,interval=-1:5,number=10]
  \tkzDrawRiemannSumInf[fill=green!40,opacity=.5,interval=-1:5,number=10]
\end{tikzpicture}
```

`\tkzDrawRiemannSumMid`[< 命令选项>]

第 1 个命令的变体，但矩形始终位于函数曲线上。

#### 9.4 `\tkzDrawRiemannSumMid`命令示例



```
\begin{tikzpicture}[scale=1.75]
\tkzInit[xmin=-3,xmax=6,ymin=-2,ymax=14,ystep=2]
\tkzDrawX \tkzDrawY
\tkzFct[line width=2pt,color = red, domain =-3:6]{(-\x-2)*(\x-5)}
\tkzDrawRiemannSumMid[fill=blue!40,opacity=.5,interval=-1:5,number=10]
\end{tikzpicture}
```

10 特殊直线

10.1 垂线

`\tkzVLine[< 命令选项>]{<decimal number>}`

由于无法使用gnuplot绘制直线，该命令需要使用fp语法定义函数。

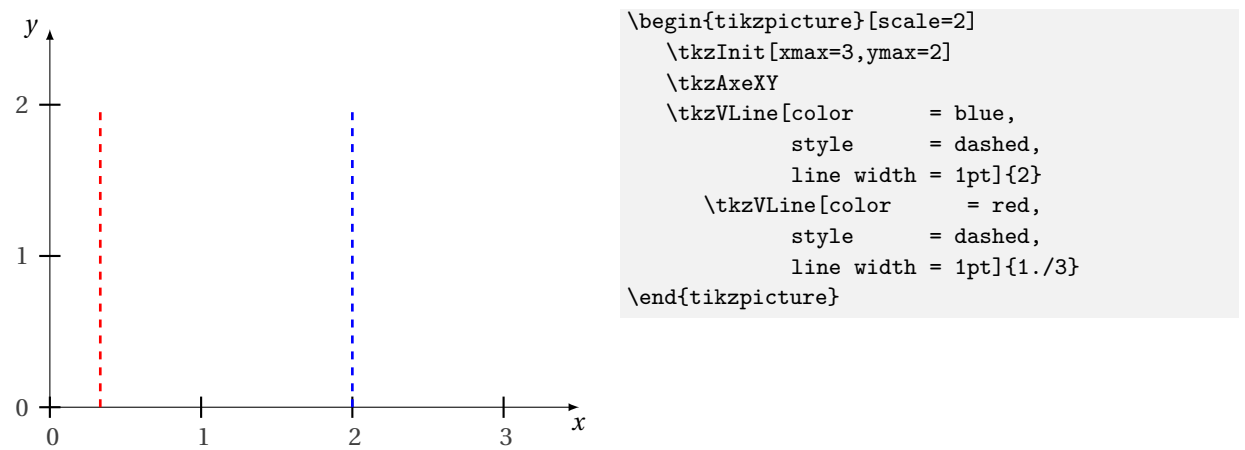
参数	样例	含义
decimal number	<code>\tkzVLine{1}</code>	垂线 $x = 1$

选项	默认值	含义
color	black	颜色
line width	0.6pt	线宽
style	solid	线型

可以使用所有有效 *TikZ* 选项。

10.2 绘制垂线

注意使用带小数点的数，以实现浮点数计算。

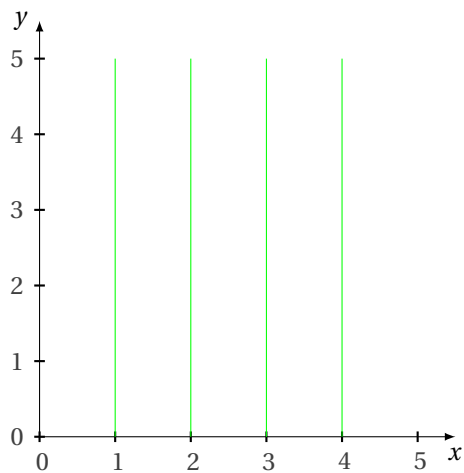


`\tkzVLines[< 命令选项>]{(list of values)}`

由于无法使用`gnuplot`绘制直线，该命令需要使用`fp`语法定义函数。

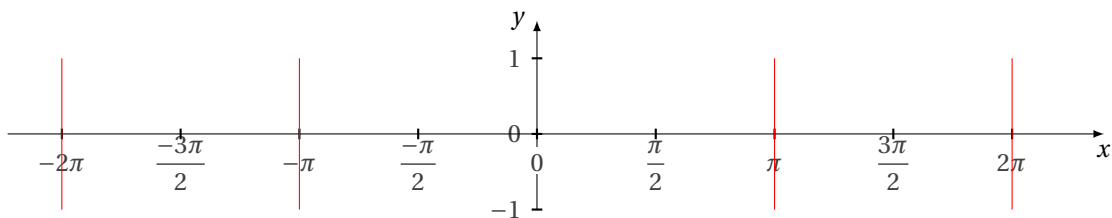
参数	样例	含义
list of values	<code>\tkzVLines{1,4}</code>	垂线 $x=1$ 和 $x=4$

### 10.3 绘制多条垂线



```
\begin{tikzpicture}
\tkzInit[xmax=5,ymax=5]
\tkzAxeXY
\tkzVLines[color = green]{1,2,...,4}
\end{tikzpicture}
```

### 10.4 用`fp`计算垂线



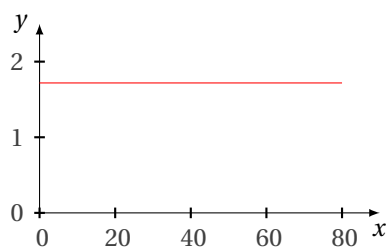
```
\begin{tikzpicture}
\tkzInit[xmin=-7,xmax=7,ymin=-1,ymax=1]
\tkzAxeY
\tkzAxeX[trig=2]
\foreach \v in {-2,-1,1,2}
{\tkzVLine[color=red]{\v*\FPpi}}
\end{tikzpicture}
```



## 10.5 水平线

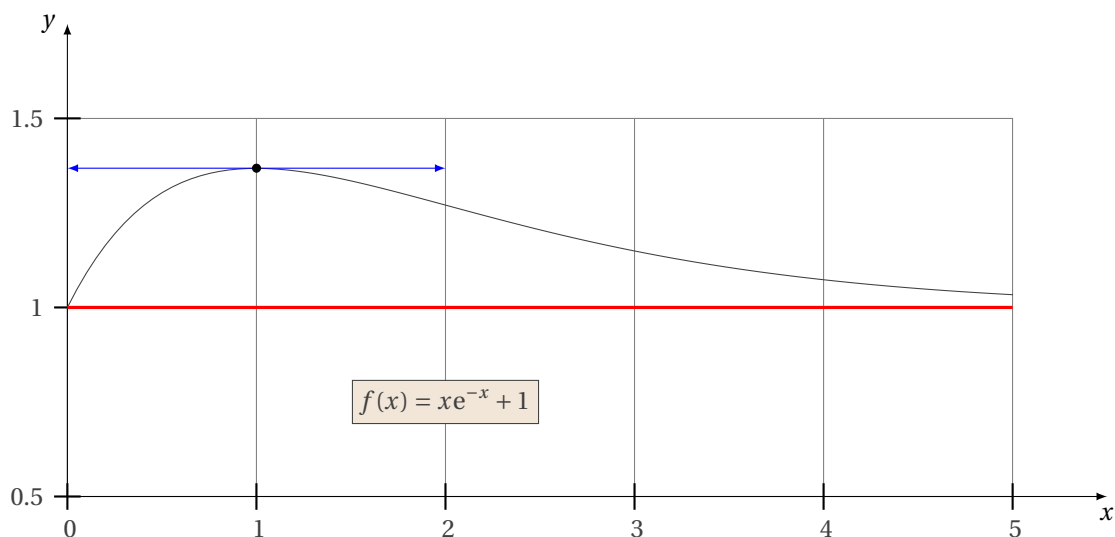
`\tkzHLine[< 命令选项>]{<decimal number>}`

参数	样例	含义
decimal number	<code>\tkzVLine{1}</code>	水平线 $y = 1$



```
\begin{tikzpicture}
\tkzInit[xmax=80,xstep=20,ymax=2]
\tkzAxeXY
\tkzHLine[color=red]{exp(1)-1}
\end{tikzpicture}
```

## 10.6 水平渐近线

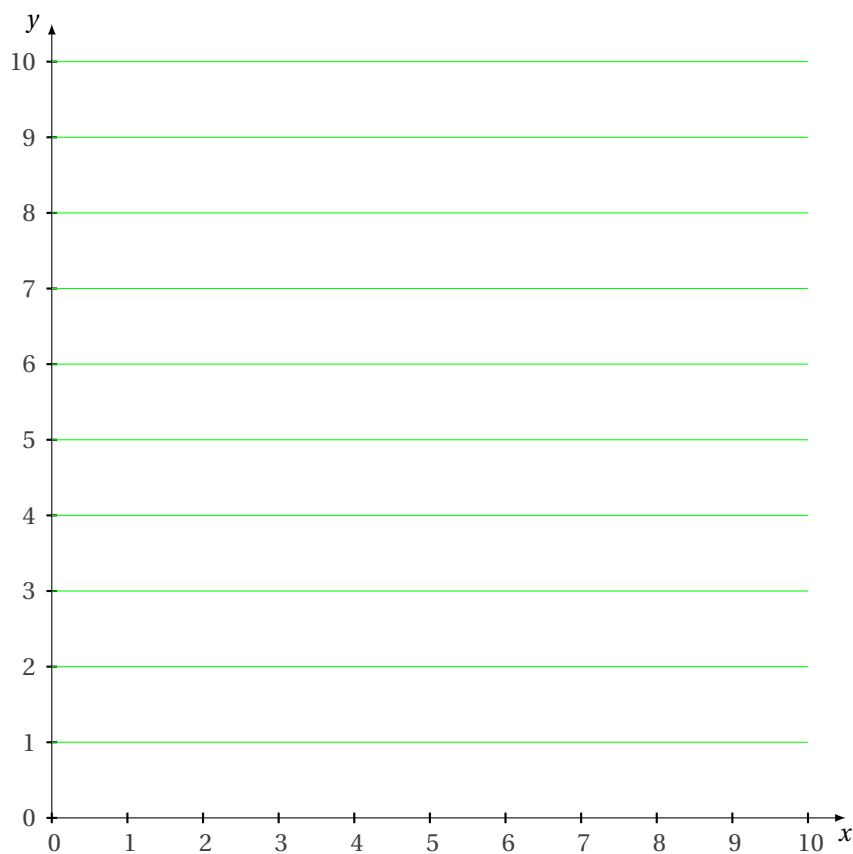


```
\begin{tikzpicture}[scale=2.5]
\tkzInit[xmax=5,ymin=0.5,ymax=1.5,ystep=0.5]
\tkzGrid
\tkzAxeXY
\tkzFct[domain = 0:10]{x*exp(-x)+1}
\tkzHLine[color=red,style=solid,line width=1.2pt]{1}
\tkzDrawTangentLine[draw,color=blue](1)
\tkzText[draw,fill = brown!20](2,0.75){$f(x)=x \text{~}\text{e}^{-x}+1$}
\end{tikzpicture}
```

## 10.7 绘制多条水平线

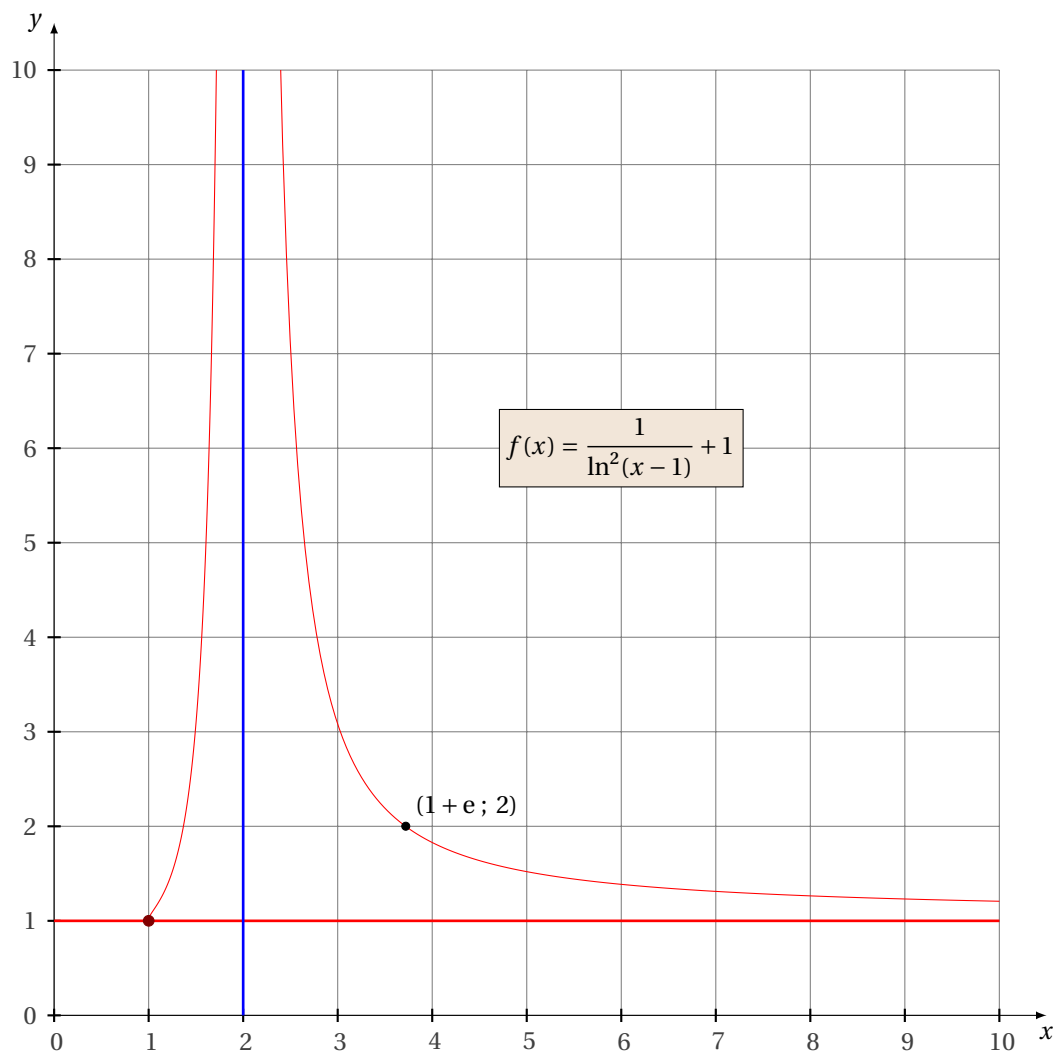
`\tkzHLines[< 命令选项>]{(list of values)}`

参数	样例	含义
list of values	<code>\tkzHLines{1,4}</code>	水平线 $y=1$ 和 $y=4$



```
\begin{tikzpicture}
\tkzInit
\tkzAxeXY
\tkzHLines[color = green]{1,2,...,10}
\end{tikzpicture}
```

## 10.8 水平和垂直渐近线



```
\begin{tikzpicture}[scale=1.25]
\tkzInit
\tkzGrid
\tkzAxeXY
\tkzFct[color=red,domain=1.001:1.9]{1+1/(log(x-1)**2)}
\tkzFct[color=red,domain = 2.1:10]{1+1/(log(x-1)**2)}
\tkzHLine[line width=1pt,color=red]{1}
\tkzVLine[line width=1pt,color=blue]{2}
\tkzDefPoint(1,1){A}
\tkzDrawPoint[fill=white,color=Maroon,size=4](A)
\tkzDefPointByFct[draw,with=b]({1+exp(1)})
\tkzLabelPoint[above right](tkzPointResult){$(1+\text{e};~2)$}
\tkzText[draw,color = black,fill = brown!20](6,6)%
    {$f(x)=\dfrac{1}{\ln^2(x-1)}+1$}
\end{tikzpicture}
```

11 参数方程的函数曲线

`\tkzFctPar[< 命令选项>]{<x(t)>}{<y(t)>}`

用gnuplot语法表示  $x(t)$  和  $y(t)$  函数， $t$  是参数变量。

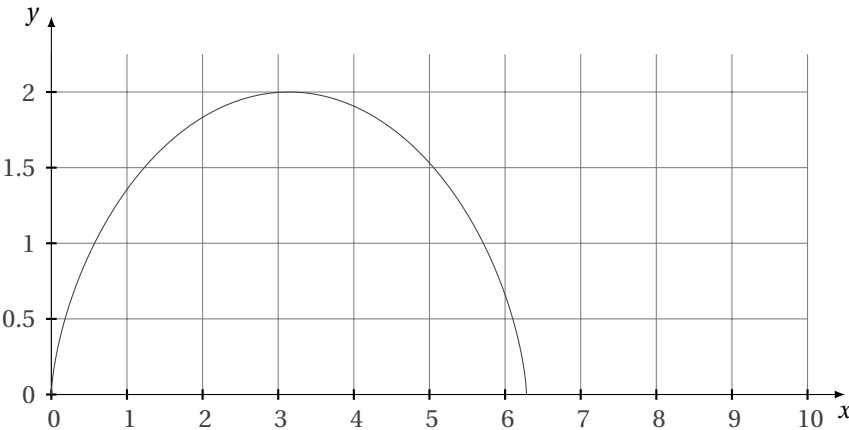
选项	样例	说明
$x(t),y(t)$	<code>\tkzFctPar[0:1]{\t**3}{\t**2}</code>	$x(t) = t^3, y(t) = t^2$

可以使用所有有效 TikZ 选项。

选项	默认值	含义
domain	-5:5	定义域
samples	200	采样点
id	tkzfονct	函数 ID
color	black	颜色
line width	0.4pt	线宽
style	solid	线型

11.1 参数曲线示例 1

$$x(t) = t - \sin(t)$$
$$y(t) = 1 - \cos(t)$$



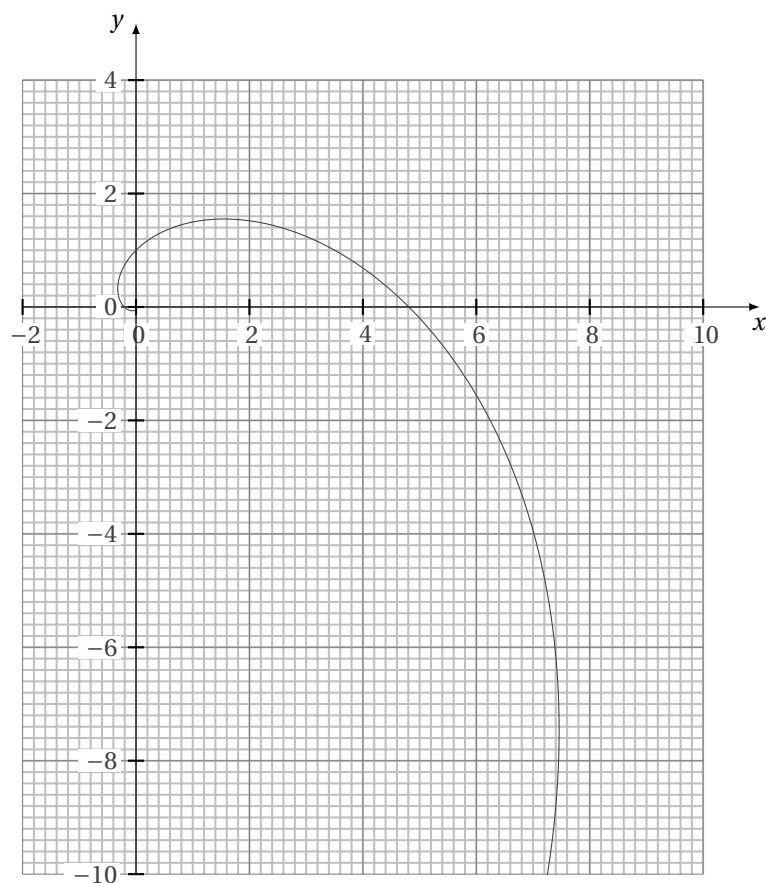
```
\begin{tikzpicture}
  \tkzInit[ymax=2.25,ystep=.5] \tkzGrid
  \tkzAxeXY
  \tkzFctPar[samples=400,domain=0:2*pi]{(t-sin(t))}{(1-cos(t))}
\end{tikzpicture}
```



## 11.3 参数曲线示例 3

$$x(t) = \exp(t) \times \sin(t)$$

$$y(t) = \exp(t) \times \cos(t)$$

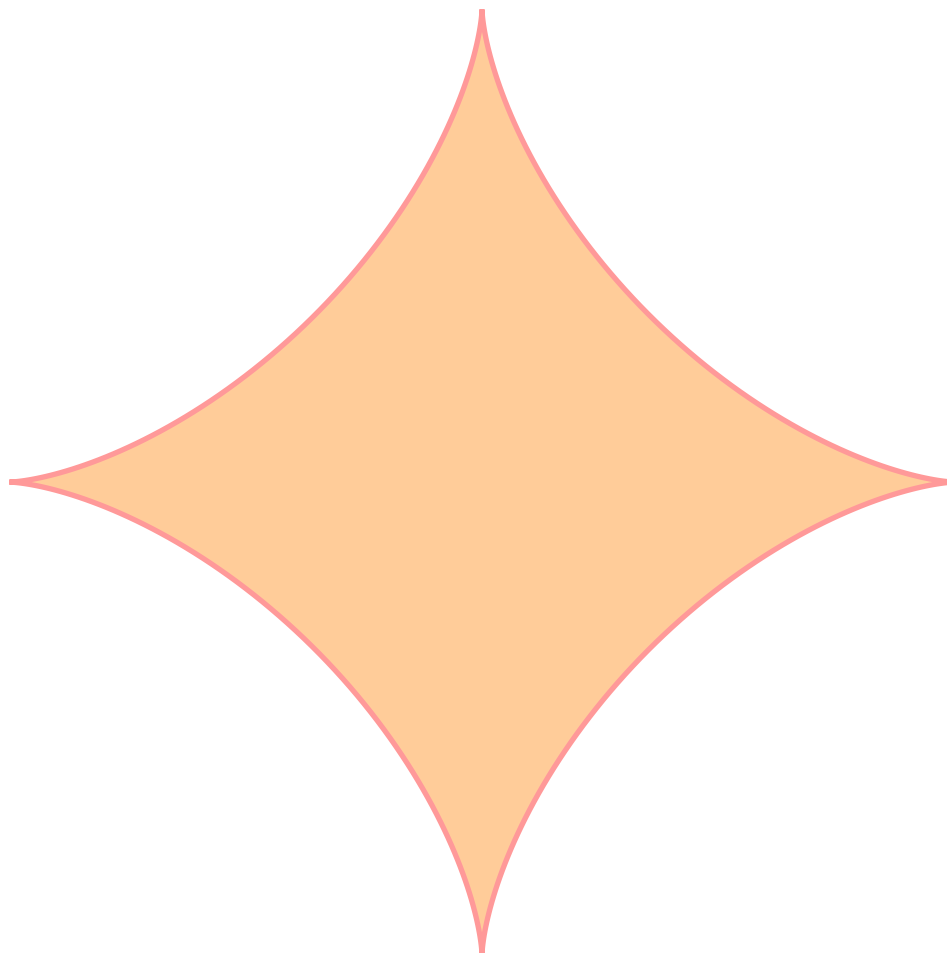


```
\begin{tikzpicture}[scale=1.5]
  \tkzInit[xmin=-2,xmax=10,xstep=2,ymin=-10,ymax=4,ystep=2]
  \tkzGrid[sub]
  \tkzAxeX[step=2]
  \tkzAxeY[step=2]
  \tkzFctPar[samples=400,domain=-pi:pi]{exp(t)*sin(t)}{exp(t)*cos(t)}
\end{tikzpicture}
```

## 11.4 参数曲线示例 4

$$x(t) = \cos^3(t)$$

$$y(t) = \sin^3(t)$$



```
\begin{tikzpicture}[scale=1.25]
  \tkzInit[xmin=-1,xmax=1,xstep=.2,
           ymin=-1,ymax=1,ystep=.2]
  \tkzFctPar[color=red,
             line width=2pt,
             fill=orange,
             opacity=.4,
             samples=400,
             domain=0:2*pi]{(cos(t))**3}{(sin(t))**3}
\end{tikzpicture}
```

## 11.5 参数曲线示例 5

心形曲线 v1

$$\begin{aligned}x(t) &= \sin^3(t) \\ y(t) &= \cos(t) - \cos^4(t)\end{aligned}$$



```
\begin{tikzpicture}[scale=4]
  \tkzInit[xmin=-1,xmax=1,ymin=-2,ymax=1]
  \tkzClip
  \tkzFctPar[samples=500,smooth,domain=-pi:pi,
    ball color=red,shading=ball]%
    {(sin(t))**3}{cos(t)-(cos(t))**4}
\end{tikzpicture}
```



## 11.6 参数曲线示例 6

心形曲线 V2, 来自<http://mathworld.wolfram.com/HeartCurve.html>

$$x(t) = \sin(t) \cos(t) \log(t)$$

$$y(t) = \sqrt{(t) \cos(t)}$$



```
\begin{tikzpicture}[scale=1.5]
  \tkzInit[xmin=-.4,xmax=.4,xstep=.1,ymin=0,ymax=.7,ystep=.1]
  \tkzClip
  \tkzFctPar[samples=2000,smooth,domain=-1:1,
    ball color=red,shading=ball]%
    {\sin(t)*cos(t)*log(abs(t))}{sqrt(abs(t))*cos(t)}
\end{tikzpicture}
```

## 11.7 参数曲线示例 7

心形曲线 V3, 来自[http://en.wikipedia.org/wiki/Heart\\_\(symbol\)](http://en.wikipedia.org/wiki/Heart_(symbol))

$$x(t) = 16 \sin^3(t)$$

$$y(t) = 13 \cos(t) - 5 \cos(2t) - 2 \cos(3t) - \cos(4t)$$



```
\begin{tikzpicture}[scale=1.75]
  \tkzInit[xmin=-20,xmax=20,xstep=5,ymin=-25,ymax=15,ystep=5]
  \tkzClip
  \tkzFctPar[samples=400,smooth,domain=0:6.28,
    ball color=red,shading=ball]%
    {16*(sin(t))**3}{13*cos(t)-5*cos(2*t)-2*cos(3*t)-cos(4*t)}
\end{tikzpicture}
```

12 极坐标函数曲线

`\tkzFctPolar[< 命令选项>]{<f(t)>}`

用gnuplot语法表示  $f(t)$ 。

参数	样例	说明
$x(t),y(t)$	<code>\tkzFctPar[0:1]{\t**3}{\t**2}</code>	$x(t) = t^3, y(t) = t^2$

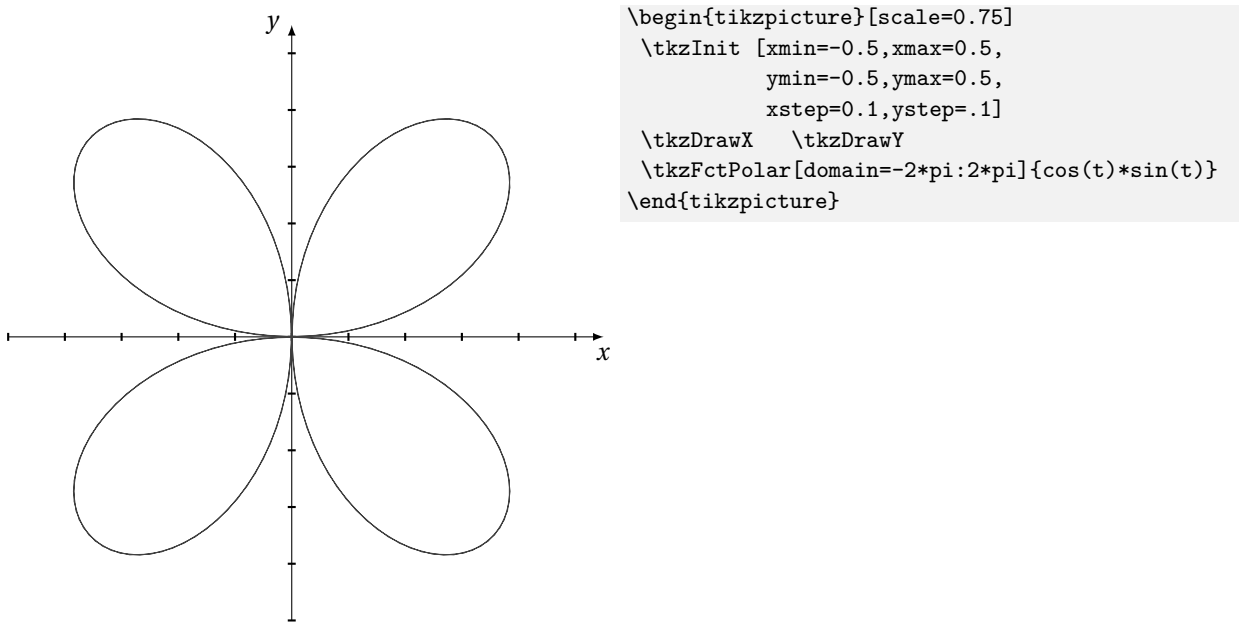
可以使用所有有效 TikZ 选项。

选项	默认值	含义
domain	0:2*pi	定义域
samples	200	采样点数
id	tkzfonct	函数 ID
color	black	颜色
line width	0.4pt	线宽
style	solid	线型

gnuplot中定义了pi表示 $\pi$ ，fp.sty中定义了\FPpi表示 $\pi$ ，值域由fp.sty宏包确定。在代码中，既可以用pi，也可以用\FPpi。

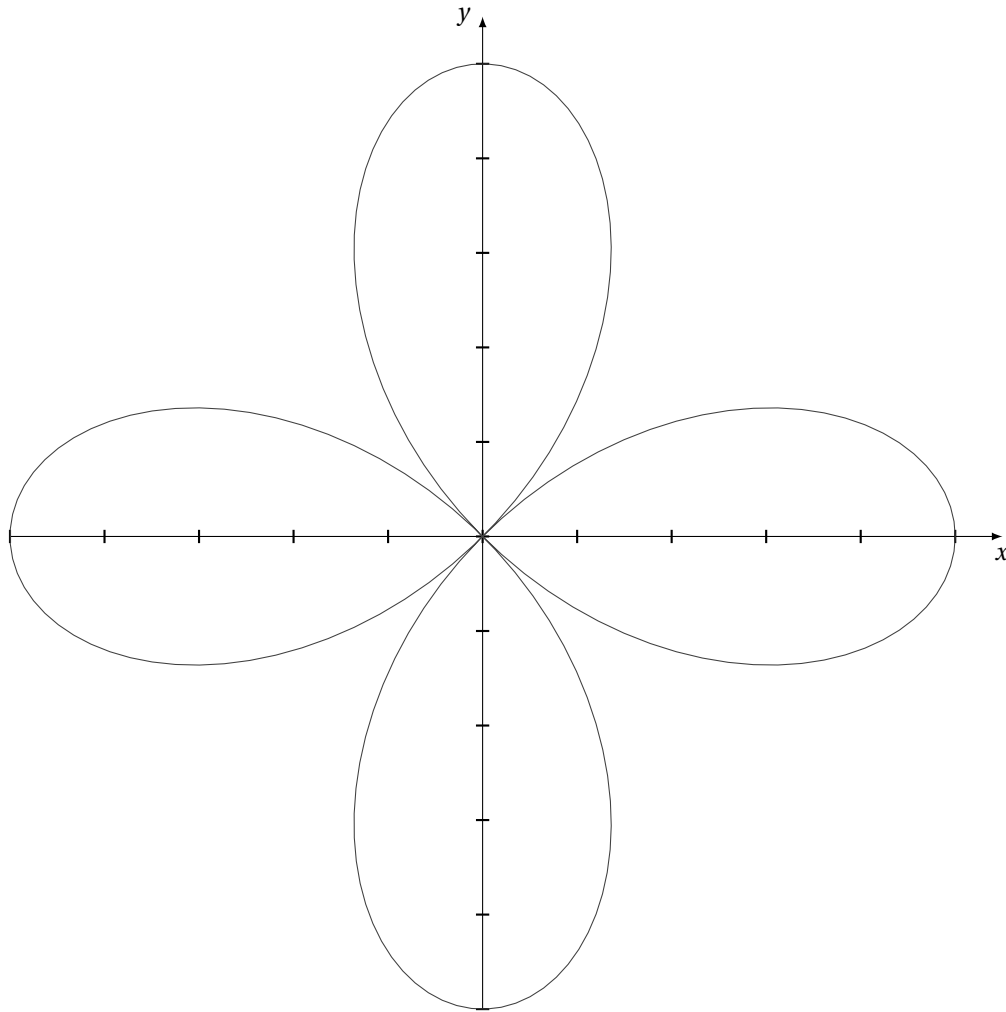
12.1 极坐标函数曲线示例 1

$\rho(t) = \cos(t) * \sin(t)$



## 12.2 极坐标函数曲线示例 2

$$\rho(t) = \cos(2 * t)$$



```
\begin{tikzpicture}[scale=1.25]
  \tkzInit [xmin=-1,xmax=1,
            ymin=-1,ymax=1,
            xstep=.2,ystep=.2]
  \tkzDrawX   \tkzDrawY
  \tkzFctPolar[domain=0:2*pi]{cos(2*t)}
\end{tikzpicture}
```

### 12.3 极坐标心形

来自: <http://mathworld.wolfram.com/HeartCurve.html>

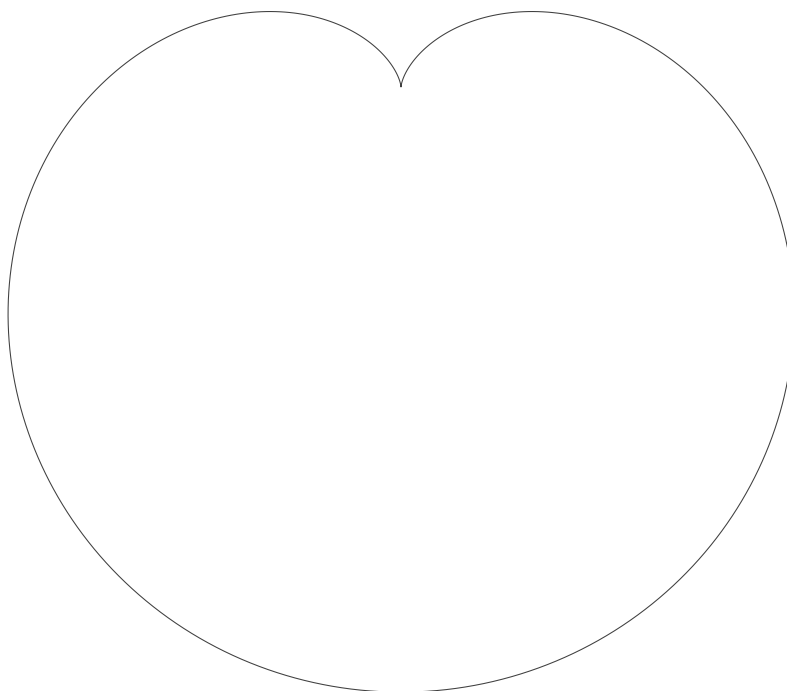
$$\rho(t) = 2 - 2 * \sin(t) + \sin(t) * \sqrt{(\cos(t)) / (\sin(t) + 1.4)}$$



```
\begin{tikzpicture}[scale=3]
\tkzInit[xmin=-5,xmax=5,ymin=-5,ymax=5]
\tkzFctPolar[domain      = -pi:pi,
             samples     = 800,
             ball color = red,
             shading      = ball]%
{2-2*sin(t)+sin(t)*sqrt(abs(cos(t)))/(sin(t)+1.4)}
\end{tikzpicture}
```

### 12.4 极坐标函数曲线示例 4

$$\rho(t) = 1 - \sin(t)$$

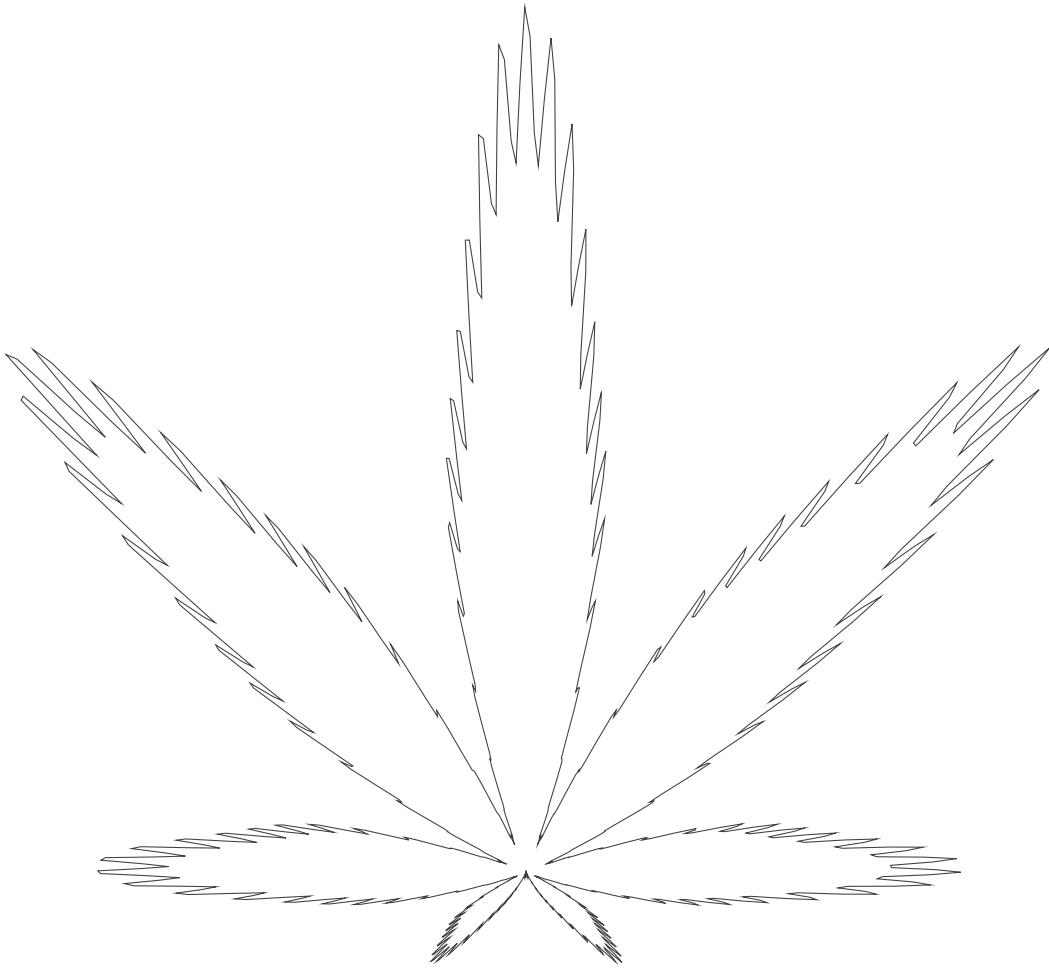


```
\begin{tikzpicture}[scale=4]
  \tkzInit [xmin=-5,xmax=5,ymin=-5,ymax=5,xstep=1,ystep=1]
  \tkzFctPolar[domain=0:2*pi,samples=400]{ 1-sin(t) }
\end{tikzpicture}
```

## 12.5 极坐标大麻曲线

来自: <http://mathworld.wolfram.com/CannabisCurve.html>

$$\rho(t) = (1 + .9 * \cos(8 * t)) * (1 + .1 * \cos(24 * t)) * (1 + .1 * \cos(200 * t)) * (1 + \sin(t))$$

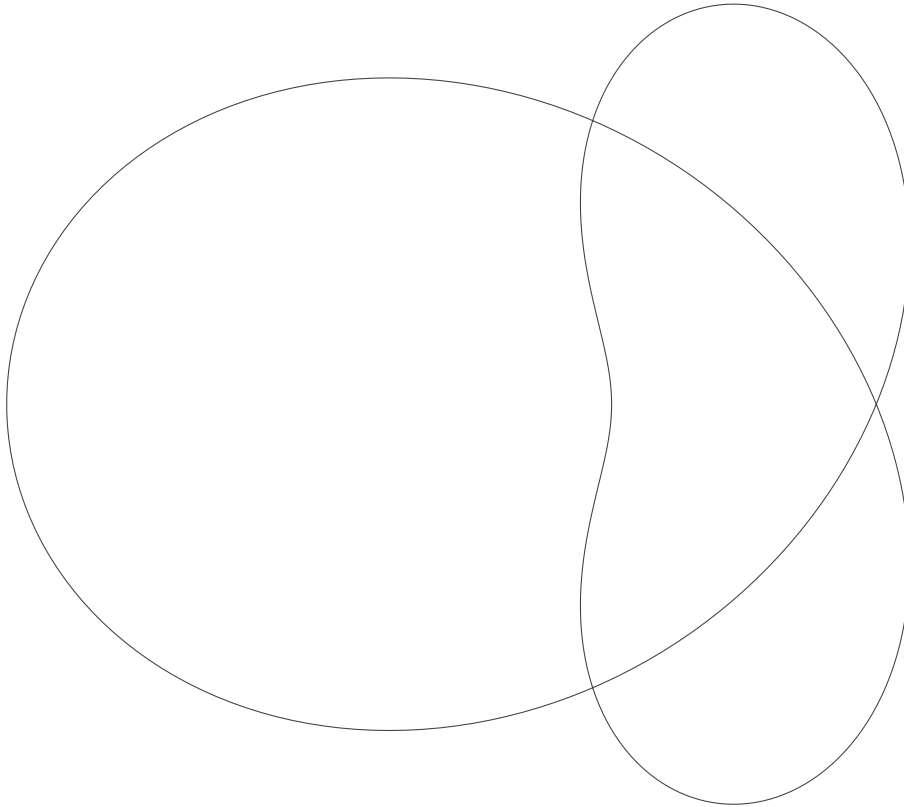


```
\begin{tikzpicture}[scale=2.5]
  \tkzInit [xmin=-5,xmax=5,ymin=-5,ymax=5,xstep=1,ystep=1]
  \tkzFctPolar[domain=0:2*pi,samples=1000]%
  { (1+.9*cos(8*t))*(1+.1*cos(24*t))*(1+.1*cos(200*t))*(1+sin(t)) }
\end{tikzpicture}
```

### 12.6 斯卡贝斯曲线

来自: <http://mathworld.wolfram.com/Scarabaeus.html>

$$\rho(t) = 1.6 * \cos(2 * t) - 3 * \cos(t)$$

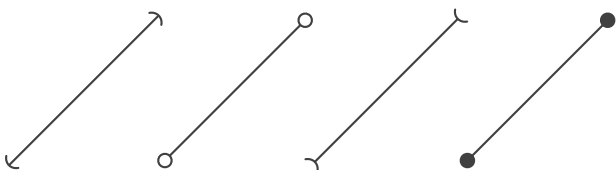


```
\begin{tikzpicture}[scale=2.5]
\tkzInit [xmin=-5,xmax=5,ymin=-5,ymax=5,xstep=1,ystep=1]
\tkzFctPolar[domain=0:2*pi,samples=400]{1.6*cos(2*t)-3*cos(t) }
\end{tikzpicture}
```



## 13 标记符号

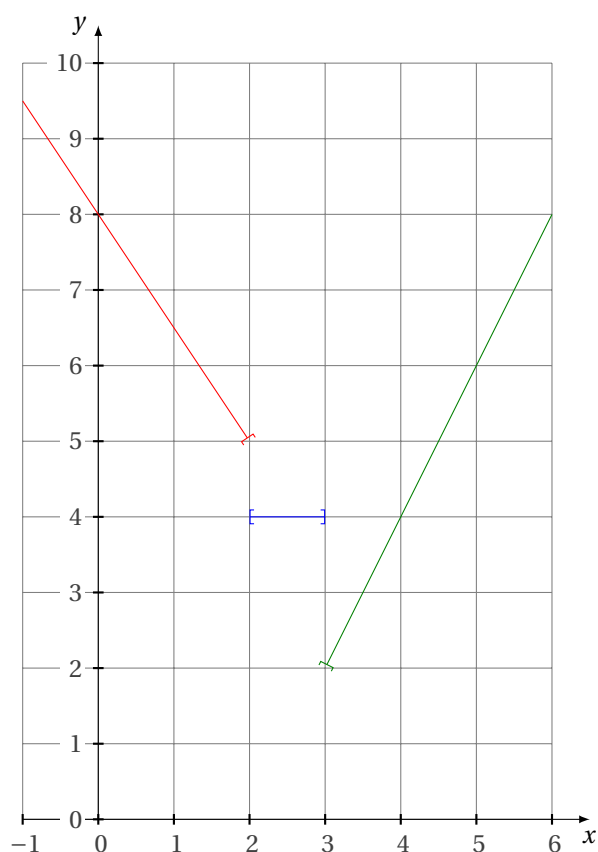
可以通过为曲线添加不同标记符号以示区分，如：



一个Simon Schläpfer的例子：

可以分别标记分段函数不同的定义域的函数。

$$y = \begin{cases} 8 - 1.5x & , \text{if } x < 2 \\ 4 & , \text{if } 2 \leq x \leq 3 \\ 2x - 4 & , \text{if } x > 3 \end{cases}$$

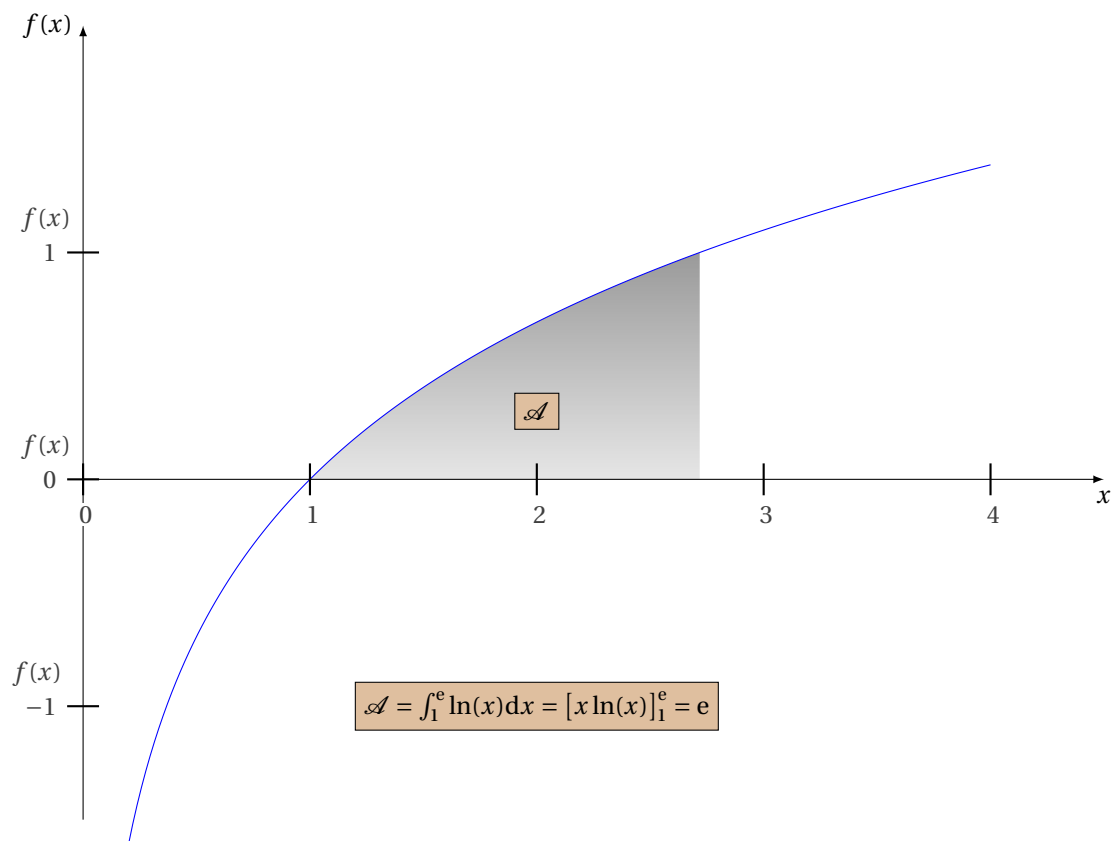


```
\begin{tikzpicture}
  \tkzInit[xmin=-1,xmax=6,ymin=0,ymax=10,xstep=1,ystep=1]
  \tkzGrid[color=gray]
  \tkzAxeXY
  \tkzFct[{-[]},color=red,domain =-1:2,samples=2]{8-1.5*\x}
  \tkzFct[{{[-]}},color=blue,domain =2:3,samples=2]{4}
  \tkzFct[[]-},color=green!50!black,domain =3:6,samples=2]{2*\x-4}
\end{tikzpicture}
```

## 14 部分实例

## 14.1 TikZ + tkz-fct实例

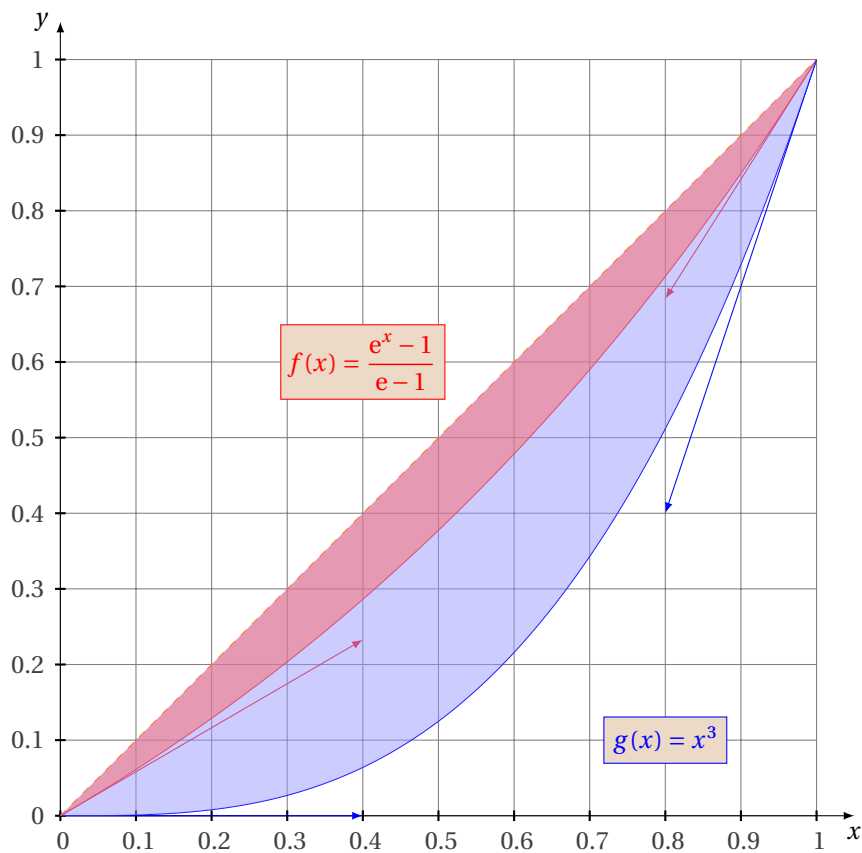
TikZ 和 **tkz-fct** 可以相互补充，因此，可以用 **tkz-fct** 处理坐标轴和文本，而用 TikZ 和 **gnuplot** 进行函数绘图。



```
\begin{tikzpicture}[scale=3]
\tkzInit[xmin=0,xmax=4,ymin=-1.5,ymax=1.5]
\tkzAxeY[label=$f(x)$]
\tkzDefPoint(1,0){x} \tkzDrawPoint[color=blue,size=0.6pt](x)
\shade[top color=gray!80,bottom color=gray!20] (1,0)%
    plot[id=ln,domain=1:2.718] function{log(x)} |- (1,0);
\draw[color=blue] plot[id=ln,domain=0.2:4,samples=200]function{log(x)};
\tkzAxeX
\tkzText[draw,color= black,fill=brown!50](2,-1)%
    {\mathcal{A} = \int_1^{\text{e}} \ln(x) \text{d}x = %
    \big[x \ln(x) \big]_1^{\text{e}} = \text{e} $}
\tkzText[draw,color= black,fill=brown!50](2,0.3){\mathcal{A}}
\end{tikzpicture}
```

## 14.2 Lorentz 曲线

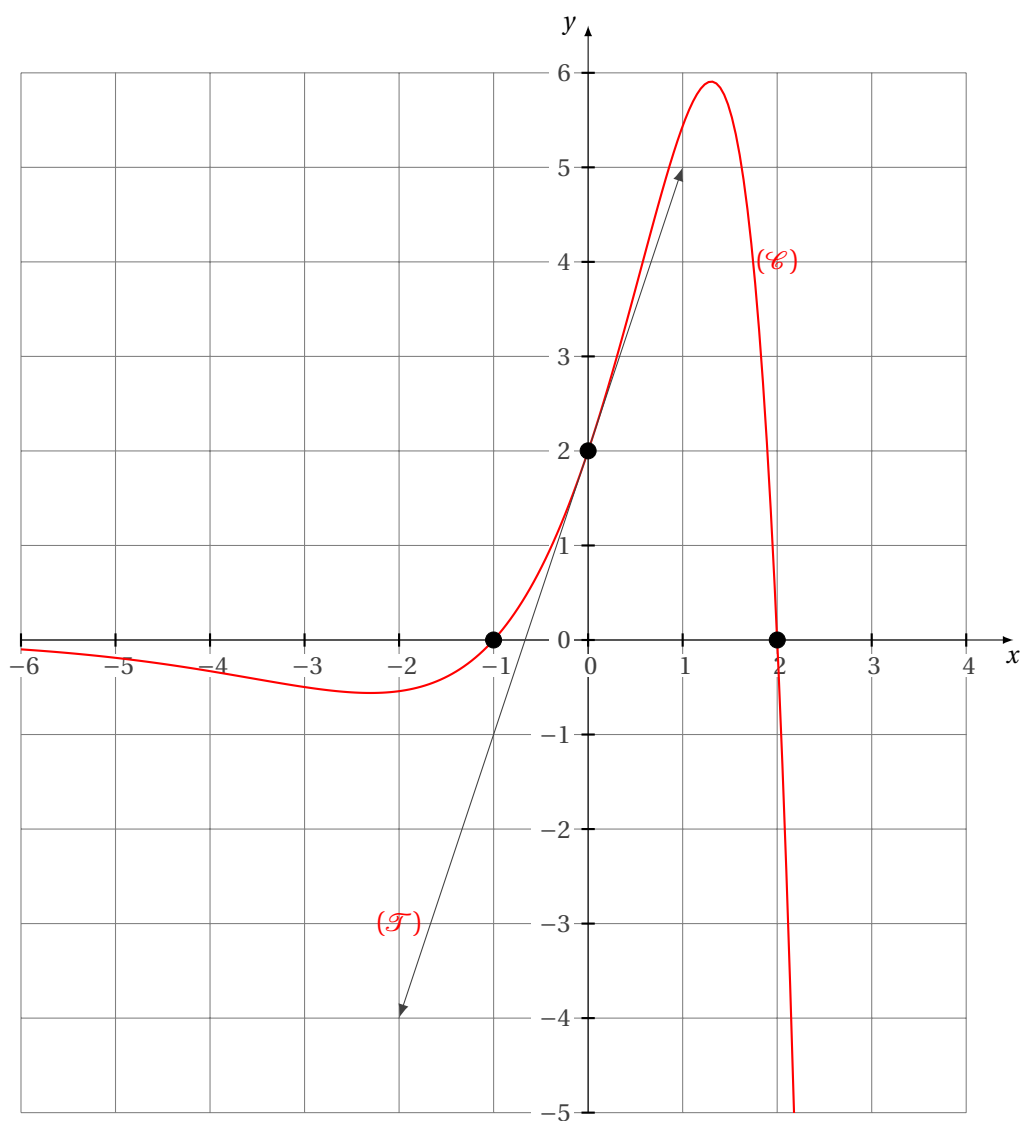
$$f(x) = \frac{e^x - 1}{e - 1} \text{ et } g(x) = x^3$$



```
\begin{tikzpicture}[scale=1]
  \tkzInit[xmax=1,ymax=1,xstep=0.1,ystep=0.1]
  \tkzGrid(0,0)(1,1)
  \tkzAxeXY
  \tkzFct[color = red,domain = 0:1]{(exp(\x)-1)/(exp(1)-1)}
  \tkzDrawTangentLine[kl=0,kr=0.4,color=red](0)
  \tkzDrawTangentLine[kl=0.2,kr=0,color=red](1)
  \tkzText[draw,color = red,fill = brown!30](0.4,0.6)%
    {\$f(x)=\dfrac{\text{e}^x-1}{\text{e}-1}\$}
  \tkzFct[color = blue,domain = 0:1]{\x*\x*\x}
  \tkzDrawTangentLine[kl=0,kr=0.4,color=blue](0)
  \tkzDrawTangentLine[kl=0.2,kr=0,color=blue](1)
  \tkzText[draw,color = blue,fill = brown!30](0.8,0.1){\$g(x)=x^3\$}
  \tkzFct[color = orange,style = dashed,domain = 0:1]{\x}
  \tkzDrawAreaafg[between=c and b,color=blue!40,domain = 0:1]
  \tkzDrawAreaafg[between=c and a,color=red!60,domain = 0:1]
\end{tikzpicture}
```

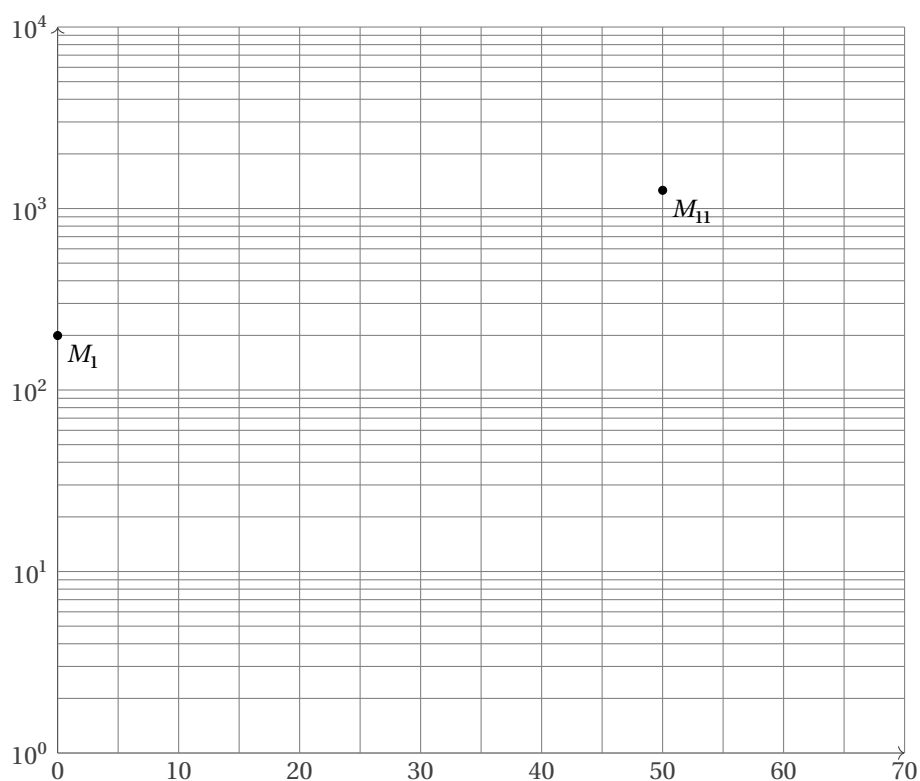
## 14.3 指数曲线

$$f(x) = (-x^2 + x + 2) \exp(x)$$



```
\begin{tikzpicture}[scale=1.25]
  \tkzInit[xmin=-6,xmax=4,ymin=-5,ymax=6]
  \tkzGrid
  \tkzAxeXY
  \tkzFct[color=red,thick,domain=-6:2.1785]{(-x*x+x+2)*exp(x)}
  \tkzSetUpPoint[size=6]
  \tkzDrawTangentLine[draw,kl=2](0)
  \tkzDefPoint(2,0){b} \tkzDrawPoint(b)
  \tkzDefPoint(-1,0){c} \tkzDrawPoint(c)
  \tkzText(2,4){(\mathcal{C})}
  \tkzText(-2,-3){(\mathcal{T})}
\end{tikzpicture}
```

## 14.4 对数坐标轴

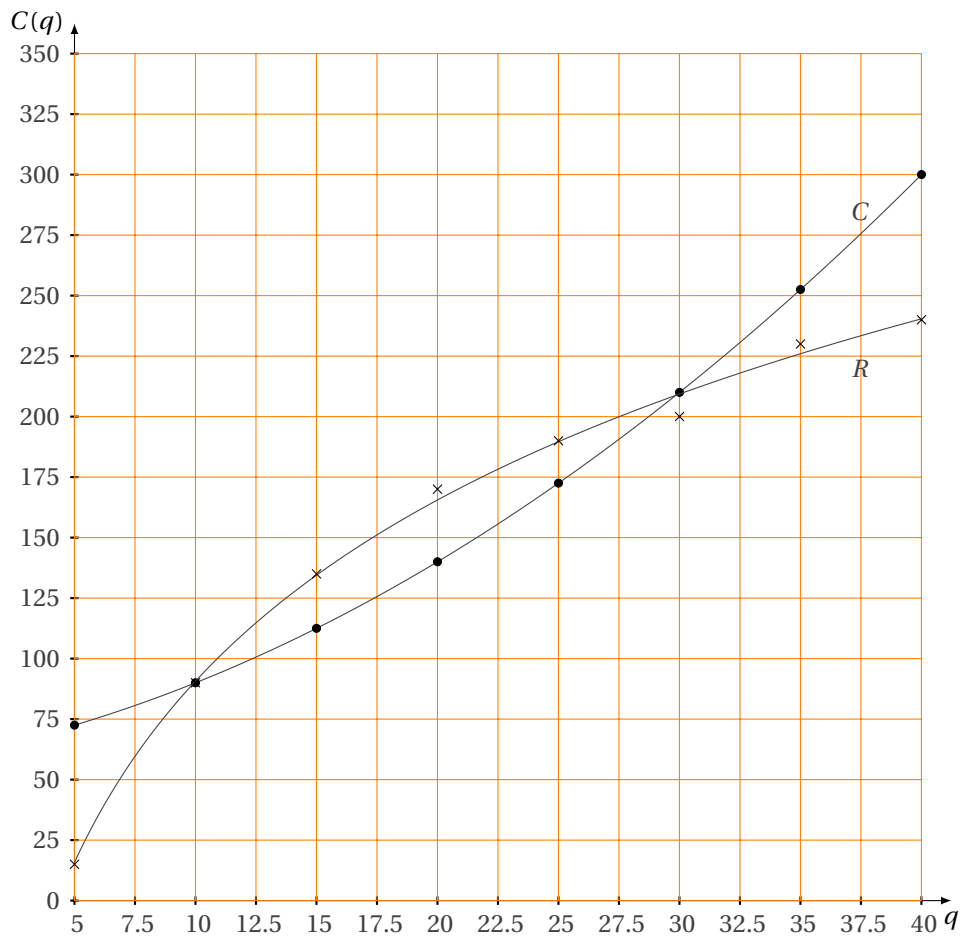


```

\begin{tikzpicture}[scale=0.8]
\tkzInit[xmax=14,ymax=12]
\draw[thin,->] (0,0) -- (14,0) node[below left] {};
\draw[thin,->] (0,0) -- (0,12) node[below left] {};
\foreach \x/\xtext in {0/0,2/10,4/20,6/30,8/40,10/50,12/60,14/70}%
{\draw[shift={(\x,0)}] node[below] {\xtext$ };}
\foreach \y/\z in {0/0,3/1,6/2,9/3,12/4}%
{\draw[shift={(0,\y)}] node[left] {$10^{\z}$};}
\foreach \x in {1,2,...,14}{\tkzVLine[gray,thin]{\x}}
\foreach \y in {3,6,...,12}{\tkzHLine[gray,thin]{\y}}
\foreach \y in {0,3,...,9}{
\foreach \z in {0.903,1.431,1.806,2.097,2.334,2.535,2.709,2.863}%
{\tkzHLine[thin,gray,shift={(0,\y)}] {\z}}}
\tkzDefPoint(0,6.90){a}
\tkzDefPoint(10,9.30){b}
\tkzDrawPoints(a,b)
\tkzLabelPoint(a){$M_{1}$}
\tkzLabelPoint(b){$M_{11}$}
\end{tikzpicture}

```

## 14.5 两条函数曲线



```

\begin{tikzpicture}[scale=.8]
\tkzInit[xmin=5,xmax=40,ymin=0,ymax=350,xstep=2.5,ystep=25]
\tkzDrawX[label=$q$]
\tkzDrawY[label=$C(q)$]
\tkzLabelXY
\tkzGrid[color=orange]
\tkzFct[domain=5:40]{0.1*\x**2+2*\x+60}
\foreach \vv in {5,10,...,40}{%
\tkzDefPointByFct(\vv)
\tkzDrawPoint(tkzPointResult)}
\tkzFct[domain=5:40]{(108*\log(\x)-158)}
\tkzText(37.5,285){$C$}
\tkzText(37.5,220){$R$}
\tkzDefSetOfPoints{%
5/15,10/90,15/135,20/170,25/190,30/200,35/230,40/240}
\tkzDrawSetOfPoints[mark = x,mark size=3pt]
\end{tikzpicture}

```

### 14.6 函数插值

例如，需要在  $[-1; 1]$  找到对如下多项式  $f$  进行插值逼近：

$$f(x) = \frac{1}{1 + 8x^2}$$

插值多项式为拉格朗日多项式：

$$\begin{aligned} P(x) = & 1.000000000 - 0.0000000072x - 7.991424876x^2 + 0.000001079x^3 + 62.60245358x^4 \\ & - 0.00004253x^5 - 444.2347594x^6 + 0.0007118x^7 + 2516.046396x^8 - 0.005795x^9 \\ & - 10240.01777x^{10} + 0.025404x^{11} + 28118.29594x^{12} - 0.05934x^{13} - 49850.83249x^{14} \\ & + 0.08097x^{15} + 54061.87086x^{16} - 0.055620x^{17} - 32356.67279x^{18} + 0.015440x^{19} \\ & + 8140.046421x^{20} \end{aligned}$$

根据霍纳的定义，使用了 21 个点进行插值，多项式的度定义为 20。下面的函数图形中，红色曲线是用 **gnuplot** 绘制的函数  $f$  的曲线，蓝色是插值函数，黄色为插值点。

#### 14.6.1 Le code

```
\begin{tikzpicture}
\tkzInit[xmin=-1,xmax=1,ymin=-1.8,ymax=1.2,xstep=0.1,ystep=0.2]
\tkzGrid
\tkzAxeXY
\tkzFct[samples = 400, line width=4pt, color = red,opacity=.5](-1---1){1/(1+8*\x*\x)}
\tkzFct[smooth,samples = 400, line width=1pt, color = blue,domain =-1:1]%
{1.0+((((((((((((((((((((
      8140.04642)*\x
      +0.01544)*\x
      -32356.67279)*\x
      -0.05562)*\x
      +54061.87086)*\x
      +0.08097)*\x
      -49850.83249)*\x
      -0.05934)*\x
      +28118.29594)*\x
      +0.02540)*\x
      -10240.01777)*\x
      -0.00580)*\x
      +2516.04640)*\x
      +0.00071)*\x
      -444.23476)*\x
      -0.00004)*\x
      +62.60245)*\x
      +0.00000)*\x
      -7.99142)*\x
      -0.00000)*\x}
\tkzSetUpPoint[size=16,color=black,fill=yellow]
\foreach \v in {-1,-0.8,---.,1}{\tkzDefPointByFct[draw](\v)}
\end{tikzpicture}
```

注意结果图像中的**龙格**现象。

#### 14.6.2 插值曲线

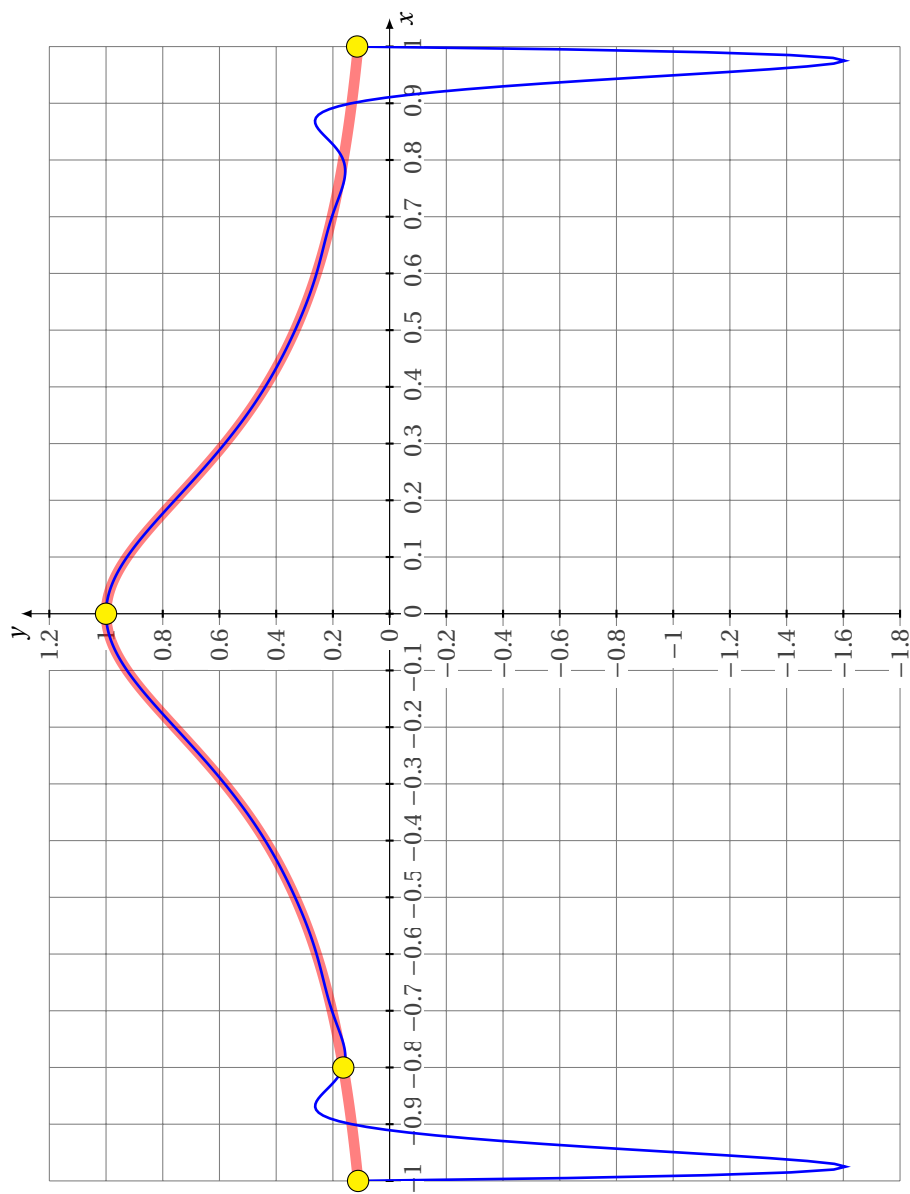


Figure 1: Interpolation :  $\frac{1}{1+8x^2}$



### 14.7 Van der Waals 曲线

令  $v$  表示体积,  $p$  表示压力。设  $b$  和  $k$  是两个正实数, Van der Waals 提出了一个表示这些之间关系的函数:

$$p(v) = \frac{-3}{v^2} + \frac{3k}{v-b}$$

其定义域为:  $I = ]b; +\infty]$

#### 14.7.1 参数关系表

$v$	$b$	$3b$	$+\infty$
$g'(v)$	0	+	0
$g(v)$	0	$\frac{8}{27b}$	0

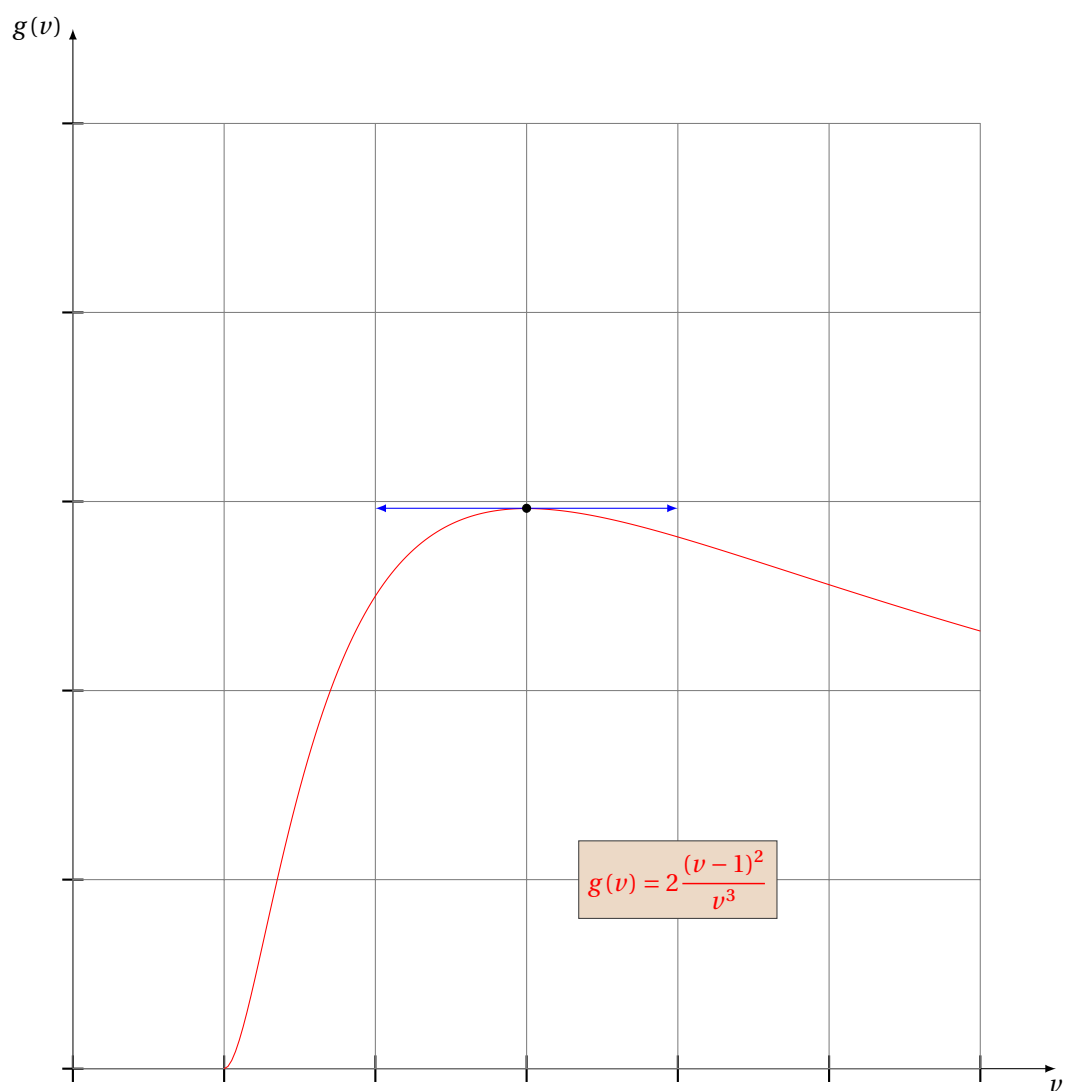
```

\begin{tikzpicture}
\tkzTab%
{ $v$ /1,%
  $g'(v)$ /1,%
  $g(v)$ /3%
}%
{ $b$ ,%
  $3b$ ,%
  $+\infty$%
}%
{0,$+,$,$0$,$-$,t}
{-/ $0$ /,%
+/$\dfrac{8}{27b}$ /,%
-/ $0$ /}%
\end{tikzpicture}

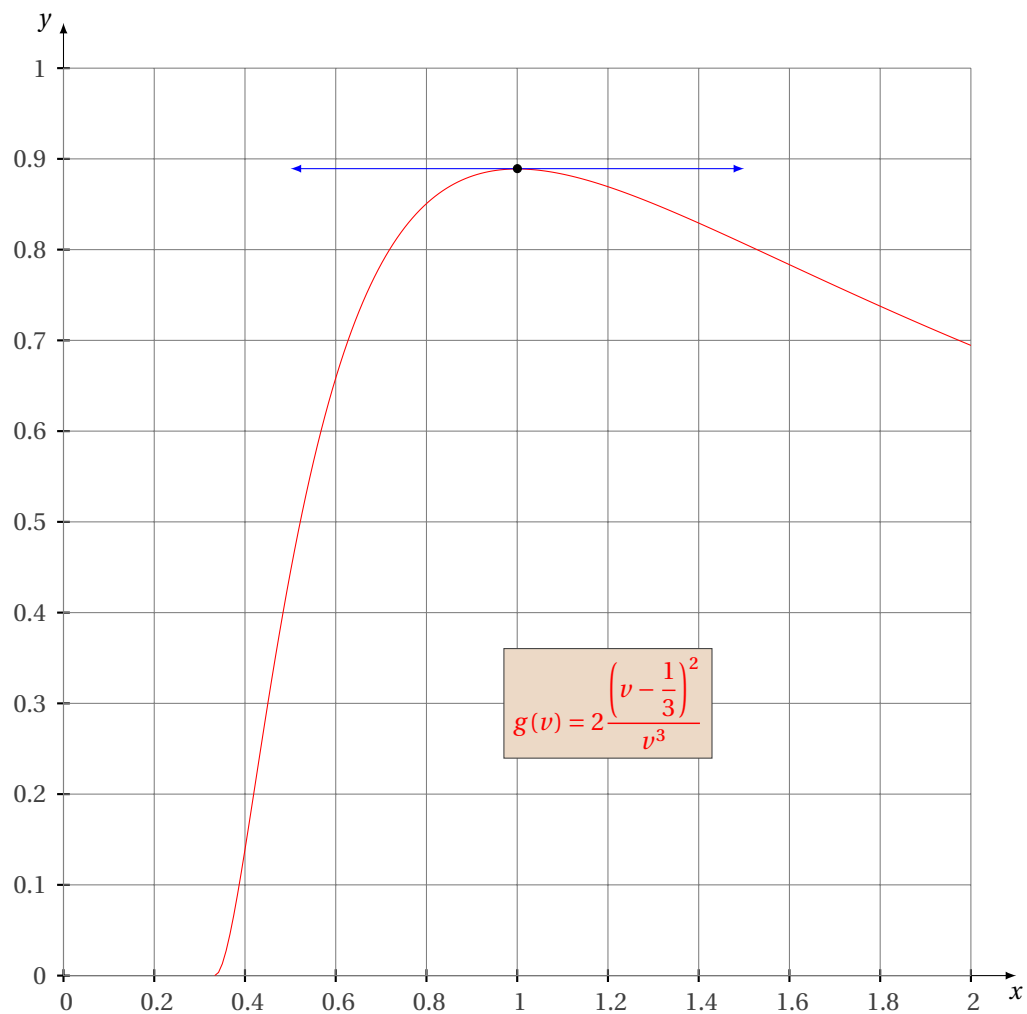
```

14.7.2  $b=1$  的第一曲线

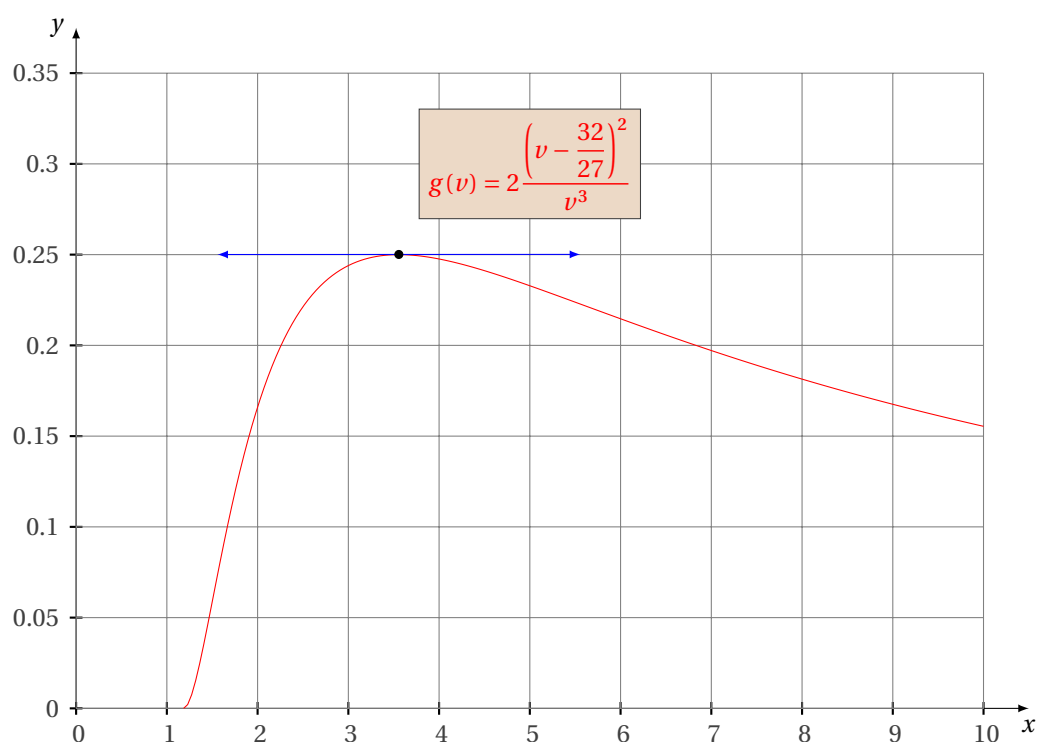
$r \leq v \leq 6$  之间的曲线族。



```
\begin{tikzpicture}[xscale=2,yscale=2.5]
  \tkzInit[xmin=0,xmax=6,ymax=0.5,ystep=0.1]
  \tkzDrawX[label=$v$]
  \tkzDrawY[label=$g(v)$]
  \tkzGrid(0,0)(6,0.5)
  \tkzFct[color = red,domain =1:6]{(2*(x-1)*(x-1))/(x*x*x)}
  \tkzDrawTangentLine[color=blue,draw](3)
  \tkzDefPointByFct(1)
  \tkzText[draw, fill = brown!30](4,0.1){$g(v)=2\dfrac{(v-1)^2}{v^3}$}
\end{tikzpicture}
```

14.7.3  $b=1/3$  的第二曲线

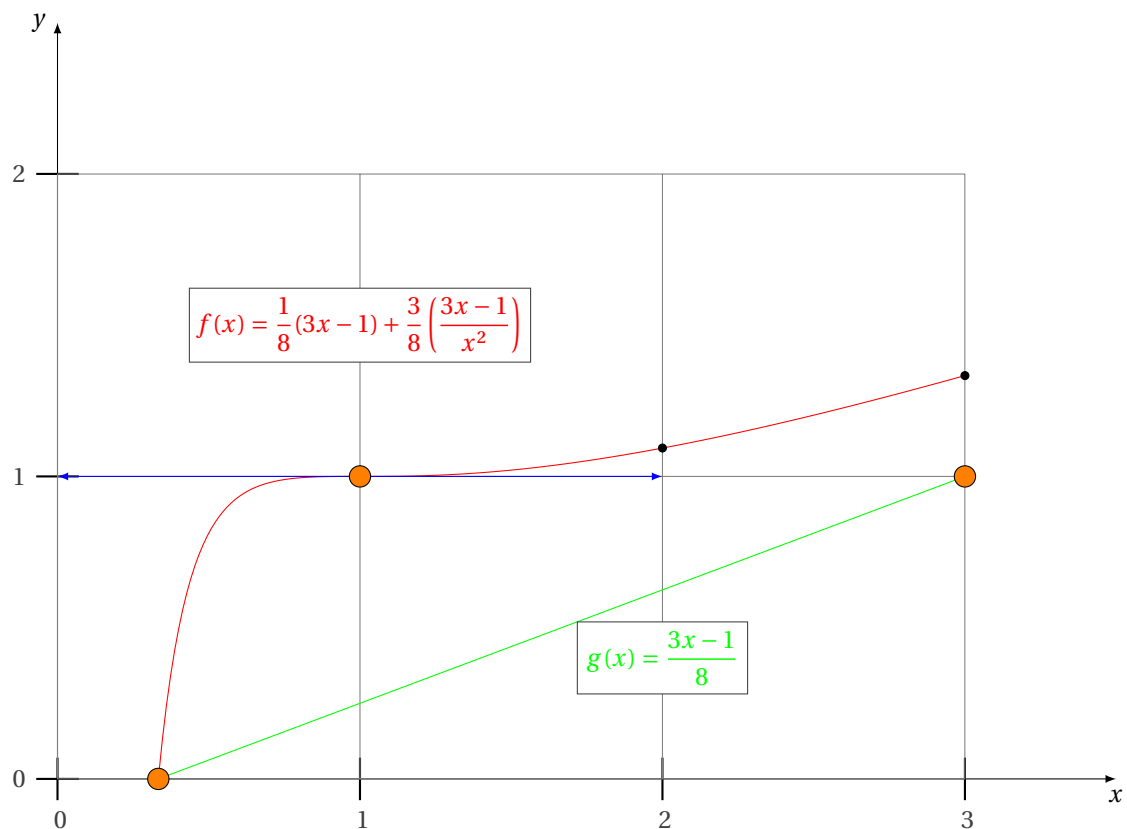
```
\begin{tikzpicture}[scale=1.2]
  \tkzInit[xmin=0,xmax=2,xstep=0.2,ymax=1,ystep=0.1]
  \tkzAxeXY
  \tkzGrid(0,0)(2,1)
  \tkzFct[color = red,domain =1/3:2]{(2*(\x-1./3)*(\x-1./3))/(\x*\x*\x)}
  \tkzDrawTangentLine[draw,color=blue,kr=.5,kl=.5](1)
  \tkzDefPointByFct(1)
  \tkzText[draw,fill = brown!30](1.2,0.3)%
    {$g(v)=2\frac{\left(v-\frac{1}{3}\right)^2}{v^3}$}
\end{tikzpicture}
```

14.7.4  $b=32/27$  的第 3 曲线

```
\begin{tikzpicture}[scale=1.2]
  \tkzInit[xmin=0,xmax=10,ymax=.35,ystep=0.05];
  \tkzAxeXY
  \tkzGrid(0,0)(10,.35)
  \tkzFct[color = red,
    domain = 1.185:10]{(2*(\x-32./27)*(\x-32./27))/(\x*\x*\x)}
  \tkzDrawTangentLine[draw,color=blue,kr=2,kl=2](3.555)
  \tkzText[draw,fill = brown!30](5,0.3)%
    {$g(v)=2\frac{\left(v-\frac{32}{27}\right)^2}{v^3}$}
\end{tikzpicture}
```

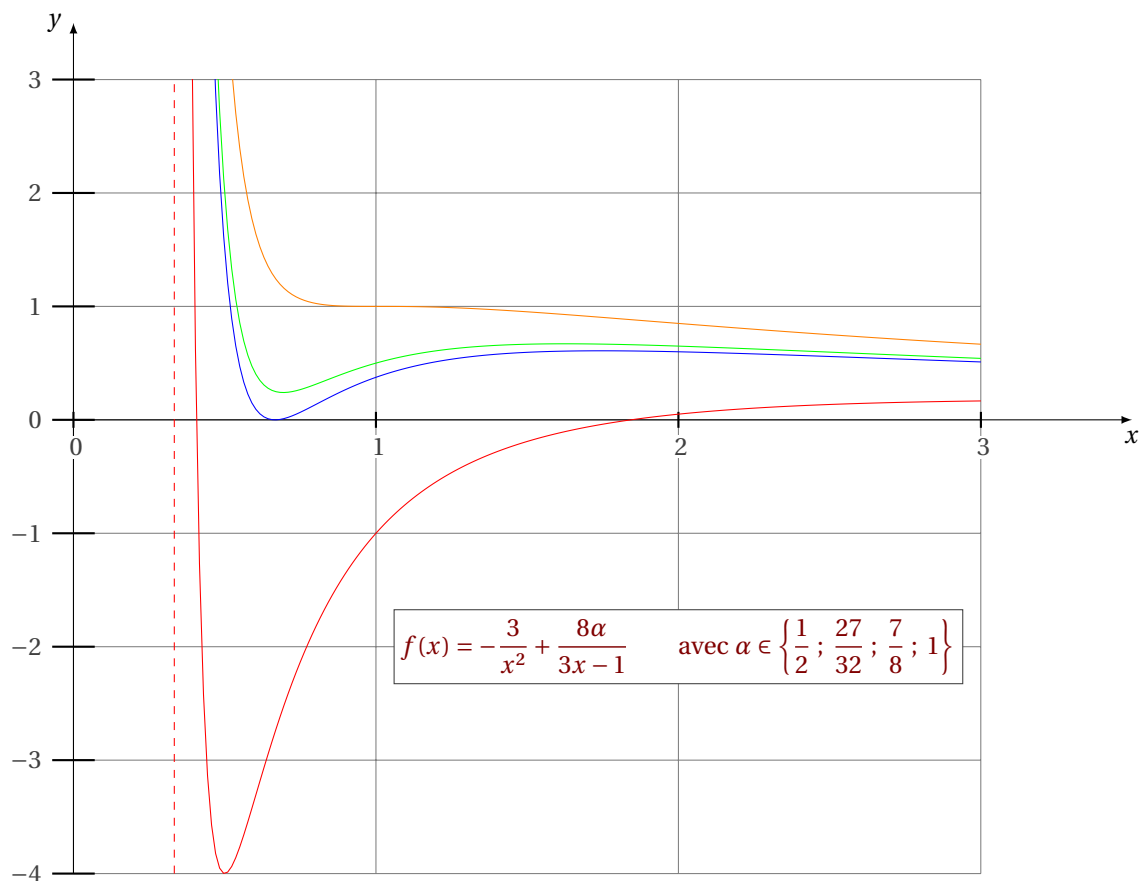
## 14.8 临界值

## 14.8.1 Van der Walls 曲线



```
\begin{tikzpicture}[scale=4]
  \tkzInit[xmax=3,ymax=2];
  \tkzAxeXY
  \tkzGrid(0,0)(3,2)
  \tkzFct[color = red,domain =1/3:3]{0.125*(3*\x-1)+0.375*(3*\x-1)/(\x*\x)}
  \tkzDefPointByFct[draw](2)
  \tkzDefPointByFct[draw](3)
  \tkzDrawTangentLine[draw,color=blue](1)
  \tkzFct[color = green,domain =1/3:3]{0.125*(3*\x-1)}
  \tkzSetUpPoint[size=8,fill=orange]
  \tkzDefPointByFct[draw](3)
  \tkzDefPointByFct[draw](1/3)
  \tkzDefPoint(1,1){f}
  \tkzDrawPoint(f)
  \tkzText[draw,fill = white,text=red](1,1.5){
    {\$f(x)=\dfrac{1}{8}(3x-1)+\dfrac{3}{8}\left(\dfrac{3x-1}{x^2}\right)\$}
  }
  \tkzText[draw,fill = white,text=green](2,0.4){\$g(x) = \dfrac{3x-1}{8}\$}
\end{tikzpicture}
```

## 14.8.2 Van der Waals 曲线 (续)

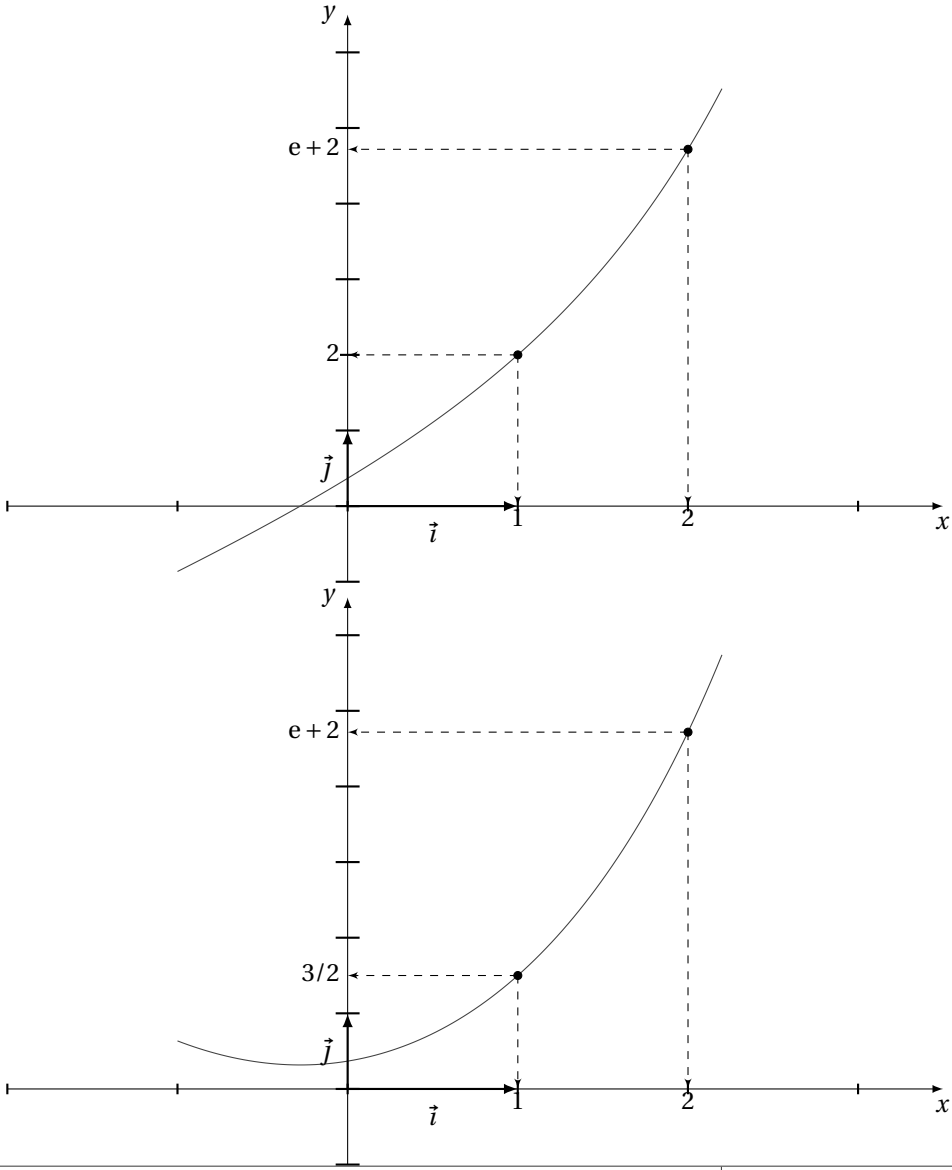


```

\begin{tikzpicture}[xscale=4,yscale=1.5]
  \tkzInit[xmin=0,xmax=3,ymax=3,ymin=-4]
  \tkzGrid(0,-4)(3,3)
  \tkzAxeXY
  \tkzClip
  \tkzVLine[color=red,style=dashed]{1/3}
  \tkzFct[color=red,domain = 0.35:3]{-3/(x*x) +4/(3*x-1)}
  \tkzFct[color=blue,domain = 0.35:3]{-3/(x*x) +27/(4*(3*x-1))}
  \tkzFct[color=orange,domain = 0.35:3]{-3/(x*x) +8/(3*x-1)}
  \tkzFct[color=green,domain = 0.35:3]{-3/(x*x) +7/(3*x-1)}
  \tkzText[draw,fill = white,text=Maroon](2,-2)%
    {$f(x)=-\dfrac{3}{x^2}+\dfrac{8\alpha}{3x-1}$ \hspace{.5cm}%
    avec $\alpha$ \in%
    \left\{\dfrac{1}{2};~\dfrac{27}{32};~\dfrac{7}{8};~1\right\}$}
\end{tikzpicture}

```

15 在alterqcm和tkz-tab宏包中绘制函数曲线

问题	答案
<p>图 1 是函数 <math>f</math> 在 <math>\mathbf{R}^+</math> 域中的图形，图 2 是基元函数 <math>f</math> 在 <math>\mathbf{R}^+</math> 域中的图形。</p> 	<p>1. 求函数 <math>f</math> 与横轴、直线 <math>x = 1</math> 和 <math>x = 2</math> 围成的面积是多少?</p> <div><input type="checkbox"/> <math>e + \frac{3}{4}</math></div> <div><input type="checkbox"/> <math>e + \frac{1}{2}</math></div> <div><input type="checkbox"/> 1</div>

问题

已知  $\mathbf{R}^+$  定义域内的严格正函数  $k$  的变分表。

$x$	0	1	3	$+\infty$
$k(x)$				$+\infty$

2. 有变化表如下，请问  $\mathbf{R}^+$  定义域中的  $g$  函数的变化表是哪个？

$$g(x) = \frac{1}{k(x)}?$$

☐ 表 A

☐ 表 B

☐ 表 C

表 A

$x$	0	1	3	$+\infty$
$g(x)$				$+\infty$

表 B

$x$	0	1	3	$+\infty$
$g(x)$				$-\infty$

表 C

$x$	0	1	3	$+\infty$
$g(x)$				0

3. 设函数  $h$  是  $\mathbf{R}$  中定义的函数：  $h(x) = \mathrm{e}^x - x + 1$ 。  $\mathcal{C}$  表示正交坐标系  $O; \vec{i}; \vec{j}$  的函数  $h$ 。

☐ 方程  $y = 1$  是  $\mathcal{C}$  的渐近线

☐ 方程  $x = 0$  是  $\mathcal{C}$  的渐近线

☐ 方程  $y = -x + 1$  是  $\mathcal{C}$  的渐近线

4. 在经济学中，边际成本是一个重要的指标，并且边际成本是总成本的导数。一项研究中，边际成本用  $C_m(q)$  表示 (百万欧元)，则他与产品数量  $q$  之间的关系为：

$$C_m(q) = 3q^2 - 10q + \frac{2}{q} + 20.$$

☐  $C_r(q) = q^3 - 5q^2 + 2\ln q + 20q + 9984$

☐  $C_r(q) = q^3 - 5q^2 + 2\ln q + 20q - 6$

☐  $C_r(q) = 6q - 10 - \frac{2}{q^2}$

以下是函数  $f$  及其基元函数的两种表示形式:

### 15.0.1 第 1 种方式

```

\begin{tikzpicture}[xscale=2.25,yscale=1]
  \tkzInit[xmin=-2,xmax=3,ymin=-1,ymax=6]
  \tkzDrawX
  \tkzDrawY
  \tkzFct[samples=100,domain = -1:2.2]{x+exp(x-1)}
  \tkzDefPoint(1,2){pt1}

```

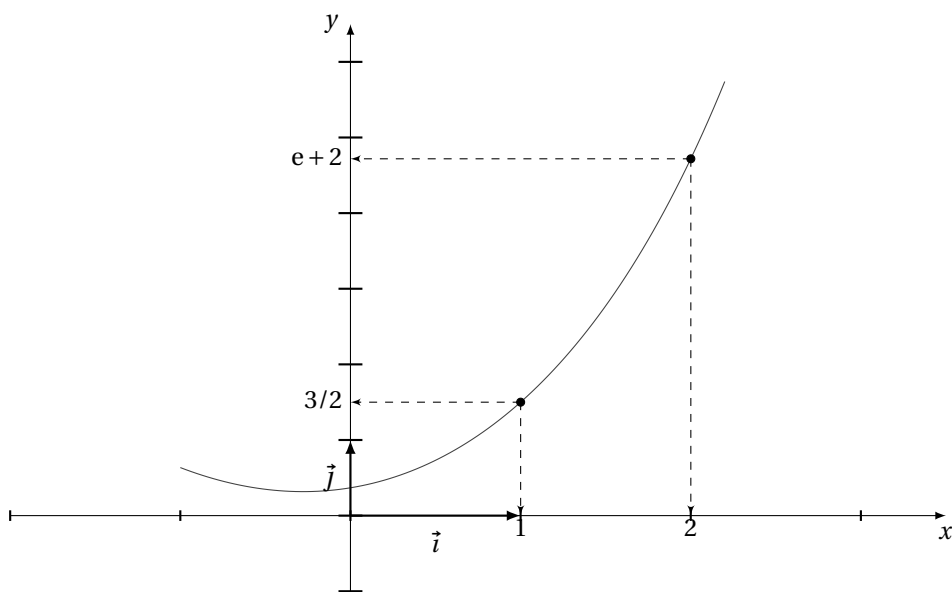


```

\tkzDrawPoint(pt1)
\tkzPointShowCoord[xlabel=$1$,ylabel=$2$](pt1)
\tkzDefPoint(2,4.71828){pt2}
\tkzDrawPoint(pt2)
\tkzPointShowCoord[xlabel=$2$,ylabel=$\text{e}+2$](pt2)
\tkzRep
\end{tikzpicture}

```

### 15.0.2 第 2 种方式



```

\begin{tikzpicture}[xscale=2.25,yscale=1]
\tkzInit[xmin=-2,xmax=3,ymin=-1,ymax=6]
\tkzDrawX
\tkzDrawY
\tkzFct[samples=100,domain=-1:2.2]{x*x/2+exp(x-1)}
\tkzDefPoint(1,1.5){pt1}
\tkzDrawPoint(pt1)
\tkzPointShowCoord[xlabel=$1$,ylabel=$3/2$](pt1)
\tkzDefPoint(2,4.71828){pt2}
\tkzDrawPoint(pt2)
\tkzPointShowCoord[xlabel=$2$,ylabel=$\text{e}+2$](pt2)
\tkzRep
\end{tikzpicture}

```

变更表代码为：

```

\begin{tikzpicture}
\tkzTabInit[lgt=1,espcl=2]{x$/0.5$,k(x)$/1.5}
{0$,1$,3$,+\infty$}
\tkzTabVar{-/ /,%
+/ /,%
-/ /,%
+/$+\infty$ /}%

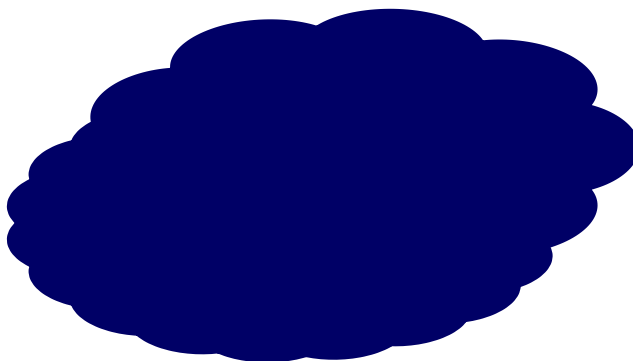
```

```
\end{tikzpicture}
```

## 16 pgfmath和fp.sty工具

### 16.1 pgfmath

即便是没有 Gnuplot，也可以绘制图形，下面是Herbert Voss(Pstricks 社区最活跃的成员) 仅使用 TikZ 绘图的一种方法。



```
\begin{tikzpicture}
\def\Asmall{0.7 } \def\Abig{3 } \def\B{20}%Herbert Voss
\path[fill=blue!40!black,domain=-pi:pi,samples=500,smooth,variable=\t]%
    plot({\Abig*cos(\t r)+\Asmall*cos(\B*\t r)},%
        {0.5*\Abig*sin(\t r)+0.5*\Asmall*sin(\B*\t r)});
\def\Asmall{0.7 } \def\Abig{3 } \def\B{10}
\path[shift={(1,1)},fill=blue!40!black,%
    domain=-pi:pi,samples=500,smooth,variable=\t]%
    plot({\Abig*cos(\t r)+\Asmall*cos(\B*\t r)},%
        {0.5*\Abig*sin(\t r)+0.5*\Asmall*sin(\B*\t r)});
\end{tikzpicture}
```

### 16.2 fp.sty

fp.sty的主要问题是可能会造成计算错误，如： $(-4)^2$  的计算，

```
\begin{tikzpicture}
\FPeval\result{(-4)^2}
\end{tikzpicture}
```

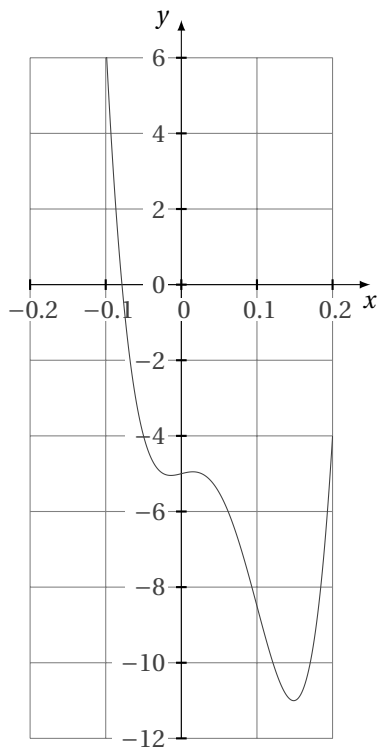
由于使用对数的方式进行计算，因此，这可能会造成错误。为此，tkz-fct.sty宏包通过修改FP@pow宏以避免这个错误。

当需要计算切线的斜率时并在函数曲线上放置点时，该宏包通过转换表达式为gnuplot提供数据，并将结果保存在tkzFcta宏中，在后续代码中，可以使用\tkzDefPointByFct和\tkzDrawTangentLine命令使用这些数据。

不过，如果需要在该图中放置一个  $x = 2$  的点，建议使用第一种方法比较合理。

否则，对于多项式函数，在使用与图像切线相关的命令时，需要将多项式表示为Horner的形式。因此，在使用\tkzFct命令时，需要将  $x^4 - 2x^3 + 4x - 5$  改写成： $-5+x*(0.5+4*x*(x*(-2+x*1)))$ 。

因此，需要做的是：



```
\begin{tikzpicture}
  \tkzInit[xmin=-0.2,xmax=0.2,xstep=.1,
    ymin=-12,ymax=6,ystep=2]
  \tkzGrid
  \tkzAxeXY
  \tkzFct[domain = -.1:.2]%
    {-5+x*(0.5+4*x*(x*(-2+x*1)))}
\end{tikzpicture}
```

## 17 注意事项

## 1. 旧版本修订:

- 将`\tkzTan`命令改为`\tkzDrawTangentLine`命令
- 现在, 在 TikZ 中使用定义域, 而不再使用  $\langle x_a \dots x_b \rangle$  的形式
- 将`\tkzFctPt`命令改为`\tkzDefPointByFct`命令

2. 当`xstep`不为 1 时, 必须在函数表达式中使用`\x`.3. 当函数表达式作为一个参数传递给`\tkzFct`命令时, 它与`\tkzFctgnu`命令中的`gnuplot`同时存储。`\tkzFctgnu`命令使用前缀“a”实现对该函数的引用, 在同一`tikzpicture`环境中, 也可以使用“b”...这样的方式按顺序对后续函数实现引用。

同时, 还按`fp.sty`语法在将前缀`tkzFcta`存储在命令`\tkzFcta`中。

最后, 所使用的宏将按`\tkzFctgnuLast`和`\tkzFctLast`两种语法保存。

4. 注意`gnuplot`的除法必须用小数点按  $1./3$  的形式计算,  $1/3$  将会按整数的方式计算, 其结果为 05. `gnuplot` 的问题:

- 如果未能创建 `xxx.table` 文件, 可能的原因有:
  - `TEX` 找不到`gnuplot`, 这多是由“PATH”环境变量设置错误引起的。
  - `TEX` 无法启动`gnuplot`, 这主要是因为未使用`shell-escape`编译参数引起的。

还有一种可能是 `xxx.gnuplot` 文件错误, 只要用文本编辑器打开它, 就可以编辑其`gnuplot`命令。值得注意的是: `gnuplot`在 4.2 版后, 语法发生了变化 (4.4 或 4.5 后会推出新讲法), 使用旧版本的创建表格命令是: `set table`。

- 在`gnuplot`定义了`pi`表示  $\pi$
- 在`fp.sty`定义了`\FPpi`表示  $\pi$ 。
- 对于直线, 设置采样点数为 2(`samples=2`) 则足以绘制该直线。

6. 幂运算  $a^b$  在 `fp` 和 `pgfmath` 中表示为  $a \wedge b$ , 在 `gnuplot` 中表示为  $a * * b$ 。7. `tkz-fct`修改了 `FP@pow`(Christian Tellechea 2009 的修订代码), 从而允许计算负数的整次幂。8.  $\{1/\exp(1)\}$  是正确的, 但  $(1/\exp(1))$  是错误的

## 17.1 gnuplot的函数

gnuplot	fp	Description
+	+	addition
-	-	soustraction
*	*	multiplication
/	/	division
**	^	exponentiation
%	absente	modulo
pi	pi	constante 3.1415
abs(x)	abs	Valeur absolue
cos(x)	cos	Arc -cosinus
sin(x)	sin	Arc -cosinus
tan(x)	tan	Arc -cosinus
acos(x)	arccos	Arc -cosinus
asin(x)	arcsin	Arc-sinus
atan(x)	arctan	Arc-tangente
atan2(y,x)	absente	Arc-tangente
cosh(x)	absente	Cosinus hyperbolique
sinh(x)	absente	Sinus hyperbolique
acosh(x)	absente	Arc-cosinus hyperbolique
asinh(x)	absente	Arc-sinus hyperbolique
atanh(x)	absente	Arc-tangente hyperbolique
besj0(x)	absente	Bessel j0
besj1(x)	absente	Bessel j1
besy0(x)	absente	Bessel y0
besy1(x)	absente	Bessel y1
ceil(x)	absente	Le plus petit entier plus grand que
floor(x)	absente	Plus grand entier plus petit que
absente	trunc(x,n)	troncature $n$ nombre de décimales
absente	round(x,n)	arrondi $n$ nombre de décimales
exp(x)	exp	Exponentielle
log(x)	ln	Logarithme népérien (base e)
log10(x)	absente	Logarithme base 10
norm(x)	absente	Distribution normale
rand(x)	random	Générateur de nombre pseudo-aléatoire
sgn(x)	absente	Signe
sqrt(x)	absente	Racine carrée
tanh(x)	absente	Tangente hyperbolique

## 18 命令列表

## 18.1 tkz-fct宏包提供的所有命令

- `\tkzFct[samples=200,domain=-5:5,color=black,id=tkzfct]{⟨gnuplot 语法函数表达式⟩}`
- `\tkzDefPointByFct[draw=false](⟨点名称⟩) -> tkzPointResult`
- `\tkzDrawTangentLine[draw=false,color=black,kr=1,kl=1,style=solid,with=a](⟨点名称⟩)`
- `\tkzDrawArea[domain=-5:5,color=lightgray,opacity=.5]`
- `\tkzArea[domain=-5:5,color = lightgray,opacity=.5]`
- `\tkzDrawAreafg[domain=-5:5,between= a and b]`
- `\tkzAreafg[domain=-5:5,between= a and b]`
- `\tkzFctPar[samples=200,domain=-5:5, line width=1pt,id=tkzfctpar] $x(t)y(t)$`
- `\tkzFctPolar[samples=200,domain=0:2*pi, line width=1pt,id=tkzfctpolar] $\rho(t)$`
- `\tkzDrawRiemannSum[interval=1:2,number=10,fill=gray]`
- `\tkzDrawRiemannSumInf [interval=1:2,opacity=.5,fill=gray]`
- `\tkzDrawRiemannSumSup [interval=1:2,number=10,fill=gray]`
- `\tkzDrawRiemannSumMid[interval=1:2,opacity=1,fill=gray]`

## 18.2 tkz-base宏包提供的命令

- `\tkzInit[xmin=0,xmax=10,xstep=1,ymin=0,ymax=10,ystep=1]`
- `\tkzAxeX`
- `\tkzDrawX`
- `\tkzLabelX`
- `\tkzAxeY`
- `\tkzDrawY`
- `\tkzLabelY`
- `\tkzGrid`
- `\tkzClip`
- `\tkzDefPoint`
- `\tkzDrawPoint`
- `\tkzPointShowCoord`
- `\tkzLabelPoint`

## Index

- `\draw plot function`, 6
- `\draw plot[id=fct] function---`., 7
- `\edef`, 21
- `\FPpi`, 59, 85
- `\global`, 21
- `\jobname`, 8
- Operating System
  - OS X, 9
  - Ubuntu Linux, 9
  - Windows, 9, 12
- `\t**2`, 52, 59
- `\t**3`, 52, 59
- TeX Distributions
  - MikTeX, 12
  - TeXLive, 11
- TikZ, 12
- `\tikzset{tan style/.style={->,>=latex}}`, 25
- `\tikzset{tan style/.style={-}}`, 25
- `\tkzArea`, 34
- `\tkzAxeX`, 87
- `\tkzAxeY`, 87
- `\tkzClip`, 87
- `\tkzDefPoint`, 87
- `\tkzDefPointByFct(0)`, 19
- `\tkzDefPointByFct`, 19, 83, 85
- `\tkzDefPointByFct`: arguments
  - decimal number, 19
- `\tkzDefPointByFct`: options
  - draw, 19
  - ref, 19
  - with, 19
- `\tkzDefPointByFct(<decimalnumber>)`, 19
- `\tkzDrawArea`, 34
- `\tkzDrawArea`: options
  - color, 34
  - domain, 34
  - opacity, 34
  - style, 34
  - with, 34
- `\tkzDrawAreafg`, 38
- `\tkzDrawAreafg`: options
  - between, 38
  - domain= min:max, 38
  - opacity, 38
- `\tkzDrawAreafg[<命令选项>]`, 38
- `\tkzDrawArea[<命令选项>]`, 34
- `\tkzDrawPoint`, 19, 87



- `\tkzDrawRiemannSum`, 43
- `\tkzDrawRiemannSum`: options
  - `interval`, 43
  - `number`, 43
- `\tkzDrawRiemannSumInf`, 44
- `\tkzDrawRiemannSumInf` [`<命令选项>`], 44
- `\tkzDrawRiemannSumMid`, 46
- `\tkzDrawRiemannSumMid` [`<命令选项>`], 46
- `\tkzDrawRiemannSumSup`, 45
- `\tkzDrawRiemannSumSup` [`<命令选项>`], 45
- `\tkzDrawRiemannSum` [`<命令选项>`] `{f(t)}`, 43
- `\tkzDrawTangentLine(0)`, 25
- `\tkzDrawTangentLine`, 25, 27, 83, 85
- `\tkzDrawTangentLine`: arguments
  - `a`, 25
- `\tkzDrawTangentLine`: options
  - `draw`, 25
  - `kl`, 25
  - `kr`, 25
  - `with`, 25
- `\tkzDrawTangentLine` [`<命令选项>`] (`<a>`), 25
- `\tkzDrawX`, 87
- `\tkzDrawY`, 87
- `\tkzFct`, 6, 14, 25, 83, 85
- `\tkzFct`: arguments
  - `gnuplot 函数表达式`, 14
- `\tkzFct`: options
  - `color`, 14
  - `domain`, 14
  - `id`, 14
  - `line width`, 14
  - `samples`, 14
  - `style`, 14
- `\tkzFcta`, 25, 85
- `\tkzFctb`, 25
- `\tkzFctgnu`, 85
- `\tkzFctgnuLast`, 85
- `\tkzFctLast`, 21, 31, 32, 85
- `\tkzFctPar[0:1]`, 52, 59
- `\tkzFctPar`, 52
- `\tkzFctPar`: arguments
  - `x(t), y(t)`, 52
- `\tkzFctPar`: options
  - `color`, 52
  - `domain`, 52
  - `id`, 52
  - `line width`, 52
  - `samples`, 52
  - `style`, 52
- `\tkzFctPar` [`<命令选项>`] `{x(t)}` `{y(t)}`, 52
- `\tkzFctPolar`, 59
- `\tkzFctPolar`: arguments
  - `x(t), y(t)`, 59

- \tkzFctPolar: options
  - color, 59
  - domain, 59
  - id, 59
  - line width, 59
  - samples, 59
  - style, 59
- \tkzFctPolar[[⟨命令选项⟩](#)]{[⟨ \$f\(t\)\$ ⟩](#)}, 59
- \tkzFctPt, 19, 85
- \tkzFct[[⟨命令选项⟩](#)]{[⟨gnuplot 函数表达式⟩](#)}, 14
- \tkzGetPoint, 19, 20
- \tkzGrid, 87
- \tkzHLine, 49
- \tkzHLine: arguments
  - decimal number, 49
- \tkzHLines{1,4}, 50
- \tkzHLines, 50
- \tkzHLines: arguments
  - list of values, 50
- \tkzHLines[[⟨命令选项⟩](#)]{[⟨list of values⟩](#)}, 50
- \tkzHLine[[⟨命令选项⟩](#)]{[⟨decimal number⟩](#)}, 49
- \tkzInit, 6, 14, 25, 87
- \tkzLabelPoint, 87
- \tkzLabelX, 87
- \tkzLabelY, 87
- \tkzPointShowCoord, 87
- \tkzSetUpPoint, 23
- \tkzTan, 85
- \tkzText, 23, 24
- \tkzVLine{1}, 47, 49
- \tkzVLine, 47
- \tkzVLine: arguments
  - decimal number, 47
- \tkzVLine: options
  - color , 47
  - line width, 47
  - style , 47
- \tkzVLines{1,4}, 48
- \tkzVLines, 48
- \tkzVLines: arguments
  - list of values, 48
- \tkzVLines[[⟨命令选项⟩](#)]{[⟨list of values⟩](#)}, 48
- \tkzVLine[[⟨命令选项⟩](#)]{[⟨decimal number⟩](#)}, 47
- \x, 6, 14, 85