zTikZ manual

Eureka

2024 年 6 月 22 日

目录

第一部分

Document

CHAPTER 1

zTikZ 宏包

目前 zTikZ 宏包能够在 Linux/Windows 下编译运行, 在 MacOS 上进行了部分的测试, 部分功能不能使用. 详细的测试信息或者是兼容情况请参见后续 Set Up 章节. 目前本宏包主要包含如下几个模块:

- gnuplot: 调用外部程序 gnuplot 进行图形的绘制
- wolfram: 调用 mathematica 或者是 wolframscript 进行计算
- python: 调用 python 进行计算, 绘图以及表格等操作
- zdraw: 目前本模块还处于测试阶段，但是目前基于 l3draw 已经能够完成基本图形的绘制
- cache: 本模块用于缓存计算结果，减少重复计算的时间. 可以应用于上述的 wolfram, python 模块

默认情况下不会加载以上的任意一个模块，在 ztikz 宏包中仅包含用于绘制点，直线，坐标轴和基本多边形的命令. 想要使用上述的模块也很简单, 在导言区使用如下命令进行加载:

```
\ztikzLoadModule{cache, gnuplot}
```

上述命令就表示加载 cache 和 gnuplot 模块. 一个值得注意的事情: 只有你加载对应的 module 时，对一个的脚本文件才会被写入项目文件夹下.

1.1 基本介绍

1.1.1 开发背景

其实还有很多基于 TikZ 并且提供和 zTikZ 宏包中绘图功能差不多宏包，这其中就有:

- 一个 TiKZ 的常见命令封装:tzplot
- 一个 3D 绘图的 TiKZ 库:tikz-3dplot
- 一个基于 PSTricks 的函数绘图包:pst-func, 可以用于绘制部分的特殊函数

用户如果觉得 zTikZ 库并不符合你的胃口，不妨试试上面的几个宏包，或者是直接使用原始的 pgf/tikz 命令进行绘图. 和 TikZ 相关的网站也是十分丰富的，以下推荐一个 TikZ 绘图的网站:TikZ Example. 这个网站中有着丰富的绘制样例并且提供了对应的绘图代码.

1.1.2 zTikZ 功能

zTikZ 宏包主要用于绘图与计算, 其中的绘图功能支持 `python`, `mathematica`, `gnuplot`, 但是这并不意味着你需要安装以上给的所有软件, 每一个软件 (模块) 之间是独立的. 你需要什么功能的, 你就在操作系统上安装对应的模块.

zTikZ 主要提供两个大功能: **绘图, 计算**. 绘图部分包括 TikZ 自身的绘图功能 (2d 部分)[1],python 的 matplotlib 绘图, 以及 mathematica 代码绘图. 计算部分包括 LaTeX3 的 xfp 宏包模块, python 的 sympy 计算模块, 以及 mathematica 模块.

注记 1.1.1 zTikZ 未来不会支持 3d 绘图部分了, 如果用于想要使用这一方面的功能, 还请使用其余的宏包或者使用其他的专业软件. 这里推荐使用 Asymptote.

1.1.3 缓存机制

zTikZ 除了提供必要的和外部程序互动的功能外, 还内置了一套 cache 系统, zTikZ 会自动把 TeX 和外部程序交互产生的结果缓存下来, 并且记录下 LaTeX 文档中调用部分源代码的 Hash 值. 如果 LaTeX 文档中的源代码 Hash 值改变, 那么 zTikZ 就会重新和外部程序交互, 重新产生结果, 并且缓存新的 Hash 值. 如果文档中的源代码的 Hash 值没有变, 那么 zTikZ 就会直接调用上一次的缓存结果. 这样做的好处是显而易见的, 就是我们不必反复的编译没有变化的内容, 直接引用缓存, 大大的减少了编译的时间.

目前 zTikZ 中的所有模块:TikZ/gnuplot[2], Python/sympy, Python/matplotlib, Mathematica 都已经实现了缓存机制. 在实际测试中, 在结果缓存后, 再次编译源文档的时间和直接插入对应数量的图片的时间几乎一致. 如果你不想使用缓存机制, 那么可以不加载 `cache` 模块.

注记 1.1.2 Known Bugs:

➠ **Hash Recover**

如果一个环境最开始的 Hash 值为 "A", 在你修改了这个环境的内容后, 使得此环境中代码的 Hash 值变为 "B". 但是如果你现在再次修改会 Hash 值为 "A" 时对应的源代码, 此刻的 Hash 值已经缓存在了 `ztikz.hash` 中, 所以再次编译时此环境对应的绘制结果并不会改变. 仍然为 Hash 值为 "B" 时的结果.

➠ **Conflict to indextool**

有可能你在启用缓存模块后, 发现编译报错 `missing \begin{document}....` 这个问题和宏包 indextool 的索引功能有关. 可以尝试先注释 `\makeindex` , `\printindex` 命令, 在图片缓存结束后, 取消注释, 然后再生成索引.

[1]由于 3d 绘图部分涉及的几个变换矩阵接口我还没想好怎么在 zTikZ 中声明, 所以目前 zTikZ 不提供 3d 绘图功能

[2]`tikzpicture` 环境或者是`\tikz` 命令生成图片的 cache 机制是依靠 tikz 的 `external` 库实现的, 感兴趣的可以去看看

1.2 Set Up zTikZ

1.2.1 兼容情况

目前 `zTikZ` 模块已经可以在 Windows 和 Linux 下正常运行, 各个平台兼容性如下:

- Windows: TEXLive 最低版本 2023
- Linux: TEXLive 最低版本 2022
- MacOS: MacTEX 由于缺少 l3sys-shell 宏包 (或者不适配), 所以并不兼容

在 Linux 平台上并没有什么需要注意的事项，重点是 Windows 平台上的兼容性; 要使得 zTikZ 正常运行，那么在引用对应的模块后，对应的系统中 (默认添加了环境变量) 须有对应的软件. 以下是部分的软件配置事项:

- sed: 模块 gnuplot, python 中部分命令基于绘图脚本中的函数以及绘制样式替换，目前采用 sed 实现, 后续可能会考虑去除 sed 依赖.
- python: 用于运行 python 脚本进行符号计算与绘图，需要 Python 库 sympy, scipy.
- wolframscript(可选): 如果需要使用 wolfram 模块的功能，那么需要安装 WolframScript 或对应软件 Mathematica. 执行命令时可以选择在云端执行, 这样就避免调用本地 Mathematica 计算内核.

1.2.2 环境配置

Linux

在 Linux 下除了 wolfram 应该都是很好安装的, 直接使用 Linux 发行版自带的包管理器即可. 在这里我提供一个在 WSL 中使用 Windows 下 Mathematica 的方法. 其实就是在 Linux 下创建一个从 Linux 到 Windows 的软连接, 如下:

```
ln -sf "/mnt/c/Program Files/Wolfram Research/WolframScript/wolframscript.exe"
/usr/bin/wolframscript
```

具体的 wolframscript 的路径根据实际情况而定.

Windows

因会使用到 gnuplot，所以在 Windows 下需要在系统环境变量中添加 gnuplot 的路径. 并且在编译此文档时需要加上对应的 `--shell-esape` 参数；在 Windows 平台，由于 TEXLive 的编译配置，需确保系统环境变量 `PATHEXT` 中已经删除 `.PY` 后缀. zTiKZ 在 Windows 下的适配是没有 Linux 下那么好的, 建议在 Linux 下使用本宏包.[3].

[3]zTiKZ 中的 Wolfram 模块可以在 Windwos 上使用并且可以输入`\`, `#`, `$` 等特殊字符, 这一点弥补了 latexalpha2 的不足

MacOS

在 MacOS 下由于缺少 l3sys-shell 宏包 (或者是不适配), 所以并不兼容. 后续我可能会考虑重 zTikZ 中用到的部分 l3sys-shell 宏包中的命令，如果此需求比较大的话. 当然，欢迎各位提 PR, 一起完善这个宏包.

1.3 绘图功能

1.3.1 Builtin TikZ

zTikZ 目前的绘图逻辑为:

- 通过 sed 修改脚本中的函数或定义域范围
- 通过 `write18` 调用 gnuplot 运行修改后的脚本
- 然后使用命令`\draw [<style>] plot file {<data>};` 绘制生成的数据文件

但是 TikZ 内部已经提供了直接调用 gnuplot 程序的命令，格式如下:

```
\draw[<draw style>] plot[<id>] function{<function>};
```

当指定内置命令中为 `<id>` 时，在 `<name>.tex` 文件的根路径下会产生两个文件; 第一个是、gnuplot 绘制的样式文件 `<name>.<id>.gnuplot`, 第二个是 gnuplot 产生的数据文件 `<name>.<id>.table`. 其中的 `<function>` 可用值可以参见表 1.1.

内置的命令也支持另外两种格式：parametric, raw gnuplot. 第一个表示绘制参数方程，第二个表示直接在文档中输入 gnuplot 绘图代码. 两者的基本格式如下：

```
% 启用参数方程绘制，默认变量为 t
\draw[<draw style>] plot[parametric, <id>] function{<function (t)>};

% 启用直接输入 gnuplot 代码绘制
\draw[<draw style>] plot[<id>] function{set samples 25; plot sin(x)};
```

目前为了几个绘图命令的统一，zTikZ 并没有采用内置的这一命令，而是直接调用了另一个外部程序 sed 用于辅助. 在后续去除 sed 依赖时可能会考虑重写这一部分命令. 但是，并不会改变已有的命令接口. 在这里给出一个这些内置命令的绘图样例:

```
\begin{tikzpicture}[domain=0:4]
    \draw[very thin,color=gray] (-0.1,-1.1) grid (3.9,3.9);
    \draw[->] (-0.2,0) -- (4.2,0) node[right] {$x$};
    \draw[->] (0,-1.2) -- (0,4.2) node[above] {$f(x)$};
    \draw[color=red] plot[id=x] function{x} node[right] {$f(x) =x$};
    \draw[color=blue] plot[id=sin] function{sin(x)} node[right] {$f(x) = \sin x$};
```

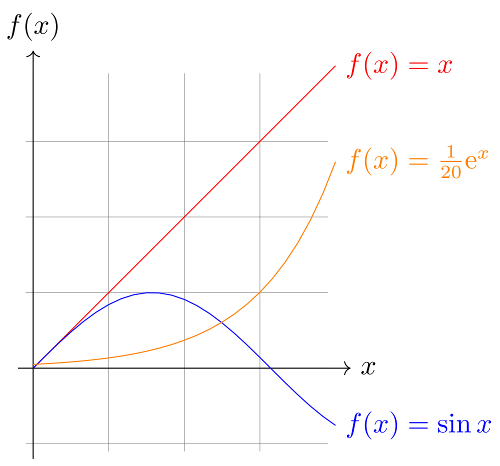


图 1.1: 12 Using Built-in TikZ Function

```
    \draw[color=orange] plot[id=exp] function{0.05*exp(x)} node[right] {$f(x) =
    \frac{1}{20} \mathrm e^x$};
\end{tikzpicture}
```

后续的统计图绘制命令也是继承自 TikZ 中内建的命令, 比如后续的`\BarPlot` 命令是如下内建命令的缩写:

```
\draw[<style>] plot[ycomb, <other style>] file{<data>};
```

更多的关于 TikZ 中这几个原生命令的使用方法请参见附录:章 三 中 TikZ 官方文档中截取的说明.

1.3.2 gnuplot

zTikZ 提供了绘制绝大部分函数的命令, 同时 zTikZ 的命令可以和 tikz 中的命令 “融合”, 它们可以在同一个 tikzpicture 环境中使用. 而且, zTikZ 对函数绘制时的坐标进行了 “对齐”. 也就是 zTikZ 中命令的坐标, 和 TiKZ 中的命令的坐标, Geogebra 中的坐标是一致的.

为何要在 zTiKZ 中把坐标 “对齐”? 试想这么一个情景: 你在 Geogebra 中找到了两个函数图像的交点为 $P(1,2)$, 首先使用 TiKZ 自带的`\filldraw` 命令把 P 点绘制出来, 然后使用 zTikZ 中 `\ShowPoint` 命令再次绘制这个 P 点. 然而结果就是: 这两个 P 点没有重合, 尽管我们指定的坐标都是 $(1,2)$ – 这就是为什么 zTikZ 要把坐标 “对齐”. 这样还有一个好处, 当你不方便使用 zTikZ 求解某些特殊的点时, 你可以在诸如 Geobebra 这样的软件中把对应的 P 点求解出来, 然后直接在 zTikZ 中使用`\ShowPoint` 命令绘制此点, 不用担心它们的坐标对齐.

在平面图形绘制方面,zTikZ 提供了绘制函数命令, 一些和坐标轴有关的命令以及部分的欧几里得几何相关的命令, 各命令[4]的名称如下:

[4]zTiKZ 中的命令基本上都遵守了 Mathematica 中的函数的命名规范

首先是和坐标系统相关的部分命令: 包括点，线，面和规则多边形的绘制以及交点的求解与绘制.

- `\ShowPoint` : 绘制点
- `\Polygon` : 绘制正多边形
- `\ShowGrid` : 绘制网格
- `\ShowAxis` : 绘制坐标轴
- `\ShowIntersection` : 绘制交点
- `\gnudata` :引用 gnuplot 数据

其次，gnuplot 模块提供了如下和函数相关的绘制命令，包括常见的几种二维函数绘制以及普通的形如 $z = f(x, y)$ 形式的三维函数的绘制:

- `\Plot` : 绘制函数
- `\ParamPlot` : 绘制参数方程
- `\ContourPlot` : 绘制等高线图
- `\PolarPlot` : 绘制极坐标图
- `\ListPlot` : 绘制散点图，只需要指定绘制参数中 `opacity=0`
- `\StairsPlot` : 绘制阶梯图
- `\StemPlot` : 绘制火柴棒图
- `\BarPlot` : 绘制条形图
- `\ShadePlot` (deprecated): 绘制渐变曲线 (直线)
- `\Plotz` : 绘制函数曲线 $z = f(x, y)$
- `\PlotPrecise` : 函数绘制精度
- `\PlotPrecise` : 函数绘制精度
- `\PlotPrecise` : 函数绘制精度

注记 1.3.1 上述的`\StairsPlot` , `\StemPlot` , `\BarPlot` , `\ShadePlot` 在绘制时只能够传入具体的数据文件，而非对应的方程.

1.3.3 Support Functions

在开始介绍之前，我想先介绍一下 gnuplot 中内建的函数，见表 1.1，摘录自：gnuplot support functions.

> Arguments to math functions in gnuplot can be integer, real, or complex unless otherwise noted. Functions that accept or return angles (e.g. sin(x)) treat angle values as radians, but this may be changed to degrees using the command set angles.

表 1.1: **Math library functions**

Function	**Arguments**	**Returns**
abs(x)	any	$\|x\|$, absolute value of x; same type
abs(x)	complex	length of x, $\sqrt{\mathrm{Re}(x)^2 + \mathrm{Im}(x)^2}$
acos(x)	any	$\cos^{-1} x$ (inverse cosine)
acosh(x)	any	$\cosh^{-1} x$ (inverse hyperbolic cosine) in radians
airy(x)	any	Airy function Ai(x)
arg(x)	complex	the phase of x
asin(x)	any	$\sin^{-1} x$ (inverse sine)
asinh(x)	any	$\sinh^{-1} x$ (inverse hyperbolic sine) in radians
atan(x)	any	$\tan^{-1} x$ (inverse tangent)
atan2(y, x)	int or real	$\tan^{-1}(y/x)$ (inverse tangent)
atanh(x)	any	$\tanh^{-1} x$ (inverse hyperbolic tangent) in radians
EllipticK(k)	real k in $(-1:1)$	$K(k)$ complete elliptic integral of the first kind
EllipticE(k)	real k in $[-1:1]$	$E(k)$ complete elliptic integral of the second kind
EllipticPi(n, k)	real $n, \|k\| < 1$	$\Pi(n, k)$ complete elliptic integral of the third kind
besj0(x)	int or real	J_0 Bessel function of x, in radians
besj1(x)	int or real	J_1 Bessel function of x, in radians
besy0(x)	int or real	Y_0 Bessel function of x, in radians
besy1(x)	int or real	Y_1 Bessel function of x, in radians
ceil(x)	any	$\lceil x \rceil$, smallest integer not less than x (real part)
cos(x)	radians	$\cos x$, cosine of x
cosh(x)	any	$\cosh x$, hyperbolic cosine of x in radians
erf(x)	any	erf(Re(x)), error function of Re(x)
erfc(x)	any	erfc(Re(x)), 1.0− error function of Re(x)
exp(x)	any	e^x, exponential function of x
expint(n, x)	any	$E_n(x)$, exponential integral function of x
floor(x)	any	$\lfloor x \rfloor$, largest integer not greater than x (real part)
gamma(x)	any	$\Gamma(\mathrm{Re}(x))$, gamma function of Re(x)

ibeta(p, q, x)	any	ibeta(Re(p, q, x)), ibeta function of Re(p, q, x)
inverf(x)	any	inverse error function Re(x)
igamma(a, x)	any	igamma(Re(a, x)), igamma function of Re(a, x)
imag(x)	complex	Im(x), imaginary part of x as a real number
invnorm(x)	any	inverse normal distribution function Re(x)
int(x)	real	integer part of x, truncated toward zero
lambertw(x)	real	Lambert W function
lgamma(x)	any	lgamma(Re(x)), lgamma function of Re(x)
log(x)	any	$\ln x$, natural logarithm (base e) of x
log10(x)	any	$\log_{10} x$, logarithm (base 10) of x
norm(x)	any	norm(x), normal distribution function of Re(x)
rand(x)	int	pseudo random number in the interval $(0:1)$
real(x)	any	Re(x), real part of x
sgn(x)	any	1 if $x > 0$, -1 if $x < 0$, 0 if $x = 0$. $\Im(x)$ ignored
sin(x)	any	$\sin x$, sine of x
sinh(x)	any	$\sinh x$, hyperbolic sine of x in radians
sqrt(x)	any	$\sqrt{x}$, square root of x
tan(x)	any	$\tan x$, tangent of x
tanh(x)	any	$\tanh x$, hyperbolic tangent of x in radians
voigt(x, y)	real	convolution of Gaussian and Lorentzian
cerf(z)	complex	complex error function
cdawson(z)	complex	complex Dawson's integral
faddeeva(z)	complex	$w(z) = \exp(-z^2) \times \mathrm{erfc}(-iz)$
erfi(x)	real	imaginary error function $\mathrm{erfi}(x) = -i \times \mathrm{erf}(ix)$
VP(x, σ, γ)	real	Voigt profile

注记 1.3.2 faddeeva(z): rescaled complex error function

1.3.4 Intruduction to Plot

我们首先来介绍上面和函数绘制相关的命令, 因为它们的参数结构几乎都是一摸一样的, 无论是参数的含义 (定义域-样式-函数), 对应参数的位置 (均为 {OOm} 形式的参数). 下面以\Plot 命令为例, 讲解这一系列 Plot 命令的参数格式以及用法:

```
\Plot[<plot domain>][<plot style>][<marker options>]{<function>}
```

其中 <plot domain> 就是绘制的定义域, 比如-3:4; <plot style> 为绘制函数的样式, 包括图形的颜色, 线型, 粗细等等;<marker option> 表示是否绘制散点图 (也许有时你需要把对应绘制命令的精度降低, 不然会十分的耗时, 或者编译结果不满意); 当你没有指定

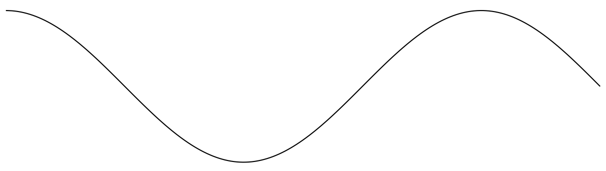
这个可选参数时，默认绘制连续函数的图像.marker 参数采用键值对的方式进行指定，见后文\ShowPoint 中的 <marker option>参数详细解释，二者用法完全一致. function 就是你要绘制的函数, 比如 sin(x). 以下为一具体示例, 首先创建一个 tikzpicture 环境，在其中写上\Plot 命令及其对应的绘制参数.

```
\begin{tikzpicture}
    \Plot[-1.5*pi:2*pi]{sin(x)}
\end{tikzpicture}
```

假如你是在命令行编译文档，那么你会看到如下的日志输出:

```
\write18 enabled.
entering extended mode
```

编译结束后，你会得到这样的一个函数图形[5].



同时在你的项目文件夹下会生成一个名为 ztikz_output 的文件夹，这个文件夹在你第一次运行\usepackage {zTikZ} 便会产生，这个文件夹用于存放 zTikZ 的缓存文件；现在我们来说说这个文件夹的结构, 当你运行了上面的\Plot 命令之后，此文件夹的结构如下 (此时会在 tikz_data 目录下生成了如图 1.2中所示的 4 个文件):

tikz_data 中的 release-figure0.pdf 即为缓存的 tikzpicture 环境的 pdf 文件，此时在对应的.md5 文件中可以看到如下内容:

```
\def \tikzexternallastkey {AE7F2539E81C96848ADCCEE3994993D1}%
```

此即保存的 tikzpicture 环境中代码的 Hash Value, 当我们改变 tikzpicture 环境中的代码时，这个 Hash value 就会改变，从而 tikz 就会再次运行此环境，重新生成图片. 这是 tikz 自带的功能，同时，zTikZ 中的 Cache 机制和此原理是十分类似的. 顺便说明命令:\gnudata 的用法 (在后面区域填充时是即为有用的):

```
\gnudata{2} = ./ztikz_output/gnuplot_data/gnu_data_1_2.table
```

\gnudata 参数中的 “2” 表示此数据是在当前 tikzpicture 环境中的第二个函数绘图数据;每一个已经绘制的函数都会在对应的文件夹下生成一个对应的数据文件，你可以使用此数据文件进行多种操作:

- 绘制此文件:\draw plot file{\gnudata {<index>}}.

[5] 自然目前这个效果我们是不满意的，没有坐标轴，网格, 刻度等元素. 后面我们会慢慢补充这幅图

```
ztikz_output
├─gnuplot_data ................................ gnuplot 缓存文件夹
│ └─gnu_data_1_1.table ....................... tikzindex-dataindex
├─mma_data ..................................... mma 缓存文件夹
├─python_data .................................. Python 缓存文件夹
├─scripts ...................................... gnuplot 绘图脚本
│ ├─contour_plot.gp
│ ├─param_plot.gp
│ ├─plot.gp
│ ├─polar_plot.gp
│ ├─sympy_script.py
│ └─float_script.py
├─tikz_data .................................... tikz 缓存文件夹
│ ├─release-figure0.dpth
│ ├─release-figure0.log
│ ├─release-figure0.md5
│ ├─release-figure0.pdf
│ └─release-figure0.run.xml
└─ztikz.hash ................................... 绘图代码 Hash
```

图 1.2: zTikZ 目录结构示意图

- 填充此文件对应曲线:`\fill [<style>] plot file{\gnudata {<index>}}`.
- 绘制此文件对应散点图: 只需在绘制散点图前指定绘制精度即可.

注记 1.3.3 目前由于技术原因, `\ContourPlot` 命令生成的数据暂时不可用于后续填充操作. 可考虑将对应隐函数转化为参数方程形式或极坐标形式, 然后再导出对应的数据. 如果你强行使用此类型数据, 那么用户可能会得到如下不良输出:

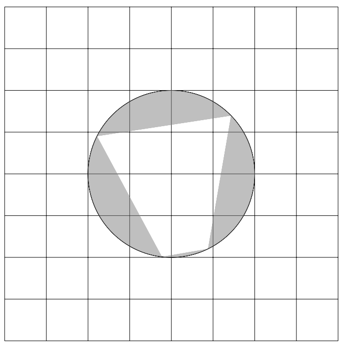
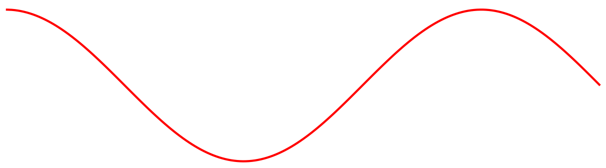


图 1.3: ContourPlot Fill Bug

后续我会解释这个文件夹中其他文件的作用, 目前我们先把函数绘制命令`\Plot` 的参数解释清楚. 如果想要设置绘制的函数图形的样式, 只需要对其第二个可选项参数进行设置即可, 比如设置为 "红色, **加粗**".

```
\begin{tikzpicture}
    \Plot[-1.5*pi:pi][red, thick]{sin(x)}
\end{tikzpicture}
```



其实上面的第二个参数的值可以是任何合法的`\draw [<plot style>]` 值, 因每一个函数绘制命令均通过如下命令实现:

```
% gnuplot data rename, plot and precise reset
\cs_new_protected:Npn \ztikz_gnu_data_plot_cs:nnn #1#2#3 {
    % rename data file
    \int_gadd:Nn \g__gnu_data_index_int {1}
    \tl_set:Nx \l__gnu_data_new_name_tl {
        gnu_data_\int_use:N \g__tikz_env_index_int _
        \int_use:N \g__gnu_data_index_int.table
    }
    \tl_set:Nx \l__gnu_data_full_path_tl
    {\g__ztikz_gnu_path_tl/\l__gnu_data_new_name_tl}
    \sys_shell_mv:xx {\g__ztikz_gnu_path_tl/gnu_data.table}
                     {\l__gnu_data_full_path_tl}
    % plot data file
    \tl_if_empty:nTF {#3}{
        \draw[#2] plot[smooth] file {\l__gnu_data_full_path_tl};
    }{
        \group_begin:
        \keys_set:nn { ztikz / point } { #3 }
        \draw[#2] plot [
            mark = \str_use:N \l__point_type_str,
            mark~ size = \dim_use:N \l__point_radius_dim,
            mark~ options = {
                rotate  = \fp_use:N \l__point_rotate_angle,
                opacity = \tl_use:N \l__point_opacity_tl,
                color   = \tl_use:N \l__point_color_tl,
                ball~ color = \tl_use:N \l__point_color_tl,
            }
        ] file {\l__gnu_data_full_path_tl};
        \group_end:
    }
    % reset precise (default 300 for plot precise)
    \bool_if:cTF {g__#1_precise_bool}{
        \PlotPrecise{#1}{100}
    }{\relax}
}
```

上述命令 `\ztikz_gnu_data_plot_cs:n` 第二个参数即为`\Plot` 命令中的第二个参数;

1.3.5 Show Axis

最后给我们的图像加上坐标轴等细节：需要用到绘制坐标轴的`\ShowAxis` 命令，绘制网格用的`\ShowGrid` 命令，以及绘制点用的`\ShowPoint` 命令.

其中`\ShowAxis` 的参数格式为:`\ShowAxis [<plot style>]{(start coordinate);(end coordinate)}`. 这里的 `<plot style>` 参数采用键值对的形式指定，可用键值对列表以及不同键的默认值可参见如下源码声明:

```
% basic tick args
tickStart       .fp_set:N   = \l__start_fp,
tickStart       .initial:n  = { -5 },
tickEnd         .fp_set:N   = \l__end_fp,
tickEnd         .initial:n  = { 5 },
axisRotate      .fp_set:N   = \l__axis_rotate_angle,
axisRotate      .initial:n  = { 0 },
% tick dimension spec
mainStep        .fp_set:N   = \l__main_step_fp,
mainStep        .initial:n  = { 1.0 },
subStep         .fp_set:N   = \l__sub_step_fp,
subStep         .initial:n  = { 0.1 },
mainTickLabel   .tl_set:N   = \l__main_tick_label_tl,
mainTickLabel   .initial:n  = { \fp_use:N {\CurrentFp} },
tickLabelShift  .dim_set:N  = \l__tick_label_shift_dim,
tickLabelShift  .initial:n  = { 0pt },
mainTickLength  .dim_set:N  = \l__main_tick_length_dim,
mainTickLength  .initial:n  = { 4pt },
subTickLength   .dim_set:N  = \l__sub_tick_length_dim,
subTickLength   .initial:n  = { 2pt },
mainTickLabelPosition .tl_set:N  = \l__main_tick_label_position_tl,
mainTickLabelPosition .initial:n = { below },
% color spec
axisColor       .tl_set:N   = \l__axis_color_tl,
axisColor       .initial:n  = { black },
mainTickColor   .tl_set:N   = \l__main_tick_color_tl,
mainTickColor   .initial:n  = { black },
subTickColor    .tl_set:N   = \l__sub_tick_color_tl,
subTickColor    .initial:n  = { black },
mainTickLabelColor .tl_set:N  = \l__main_tick_label_color_tl,
mainTickLabelColor .initial:n = { black },
% tick cross type spec
tickStyle        .choice:,
tickStyle/cross .code:n     = \tl_set:Nn \l__tick_spec_tl { cross },
tickStyle/above .code:n     = \tl_set:Nn \l__tick_spec_tl { above },
tickStyle/below .code:n     = \tl_set:Nn \l__tick_spec_tl { below },
```

一个自定义\ShowAxis 命令示例如下:

```
\NewDocumentCommand{\xAxis}{O{-2}O{8}}{
    \ShowAxis[
        tickStart=\fp_eval:n {#1+1}, tickEnd=\fp_eval:n {#2-0.75},
        mainStep=1, subStep=.25,
        axisRotate=0, axisColor=black,
        mainTickColor=black, subTickColor=black,
        mainTickLength=10pt, subTickLength=5pt,
        tickLabelShift=0pt, tickStyle=below,
        mainTickLabelPosition=below
    ]{(#1, 0); (#2, 0)}
}
```

从上述 zTikZ 内置的\xAxis 命令可以看出，我们可以指定坐标轴的如下属性:

- 主 (子) 刻度绘制起点/终点
- 主 (子) 刻度颜色设置
- 主刻度标签颜色/位置，可选位置有:above, below, left, right
- 主 (子) 刻度长度
- 主 (子) 刻度间隔
- 主刻度坐标偏移量
- 主 (子) 刻度旋转角度，请注意调整旋转后标签的位置.
- 主 (子) 刻度样式:cross, above, below, 分别代表 ticks 在坐标轴的两侧还是某一侧.

\ShowAxis 中的第二个参数表示绘制的坐标轴的起点和终点，使用 “;” 进行分割 (zTikZ 中凡是单个参数中含有多个对象的，分割对象所用到的符号都是 “;”). zTikZ 内置\xAxis ,\yAxis 命令，用于绘制两条标准的坐标轴. 命令的参数格式为:

```
\xAxis[<start>][<end>]
\yAxis[<start>][<end>]
```

上面的 <start>, <end> 分别表示 x, y 轴对应的坐标轴绘制的起始, 终止点. 对应 x 轴即为:(<start>, 0) -- (<end>, 0). y 轴即交换坐标.

注记 1.3.4 如果在使用\ShowAxis 命令时, 没有指定可选参数中键 tickStyle 的值时，那么此时并不会绘制任何的刻度.

今后用户若需要多次绘制坐标轴及其对应的标签, 可以在导言区自定义一个\ShowXYAxis 命令，自动完成坐标轴绘制以及对应的标签.

```
\newcommand{\ShowXYAxis}[2]{
    \ShowAxis{(-#1, 0); (#1, 0)}
    \ShowAxis{(0, -#2); (0, #2)}
    \ShowPoint[opacity=0]{(0, 0)}[$O$][below left]
    \ShowPoint[opacity=0]{(#1, 0)}[$x$][below]
    \ShowPoint[opacity=0]{(0, #2)}[$y$][left]
}
```

然后使用命令\ShowXYAxis 即可完成坐标轴的绘制以及对应的标注. 第一个参数:$x$ 的绘制范围为 [-#1, #1], 第二个参数:$y$ 的绘制范围为 [-#2, #2]. 默认两轴对称, 如果需要更多的样式，请使用\ShowAxis 命令进行自定义. 更多的关于坐标轴设置的样例可以参见: 节 1.12 – Plot Gallery.

在绘制完坐标轴之后，就是网格绘制; 使用\ShowGrid 命令.\ShowGrid 命令的参数也是和\ShowAxis 命令的参数一样的，只不过此命令中可以指定一个 step 关键字，用于指定绘制网格的步长 (间隔), 如 step=.5, 设置步长为 0.5.

1.3.6 Show Point

对应的\ShowPoint 命令的参数格式为:

```
\ShowPoint[<marker option>]{(coordinate 1); (coordinate 2); ...}[<label 1>; <label2>;
...][<position>]
```

上述的 <marker option>>通过 <key>-<value> 的格式进行指定, 可用的 <key>-<value> 列表为:

- <type>: zTikZ 库已经加载 pgfmarkers 库，所以任何在此库中的形状均为有效值, 默认为实心 circle 样式，即 type=*. 更多样式可以参见：图 1.4(截取自 pgf 手册).
- <radius>: dimension, 点的半径，默认为 1pt.
- <color>: color, 点的颜色, 默认为 black.
- <opacity>: float value, 点的透明度, 默认为 1，即不透明.
- <rotate>: marker 的旋转角度, 默认为 0.

终于，现在我们可以给出一个相对完整的代码 (包括 <marker option> 对应的设置):

```
\pgfuseplotmark{-}
\pgfuseplotmark{|}
\pgfuseplotmark{o}
\pgfuseplotmark{asterisk}
\pgfuseplotmark{star}
\pgfuseplotmark{10-pointed star}
\pgfuseplotmark{oplus}
\pgfuseplotmark{oplus*}
\pgfuseplotmark{otimes}
\pgfuseplotmark{otimes*}
\pgfuseplotmark{square}
\pgfuseplotmark{square*}
\pgfuseplotmark{triangle}
\pgfuseplotmark{triangle*}
\pgfuseplotmark{diamond}
\pgfuseplotmark{diamond*}
\pgfuseplotmark{halfdiamond*}
\pgfuseplotmark{halfsquare*}
\pgfuseplotmark{halfsquare right*}
\pgfuseplotmark{halfsquare left*}
\pgfuseplotmark{pentagon}
\pgfuseplotmark{pentagon*}
\pgfuseplotmark{Mercedes star}
\pgfuseplotmark{Mercedes star flipped}
\pgfuseplotmark{halfcircle}
\pgfuseplotmark{halfcircle*}
\pgfuseplotmark{heart}
\pgfuseplotmark{text}
```

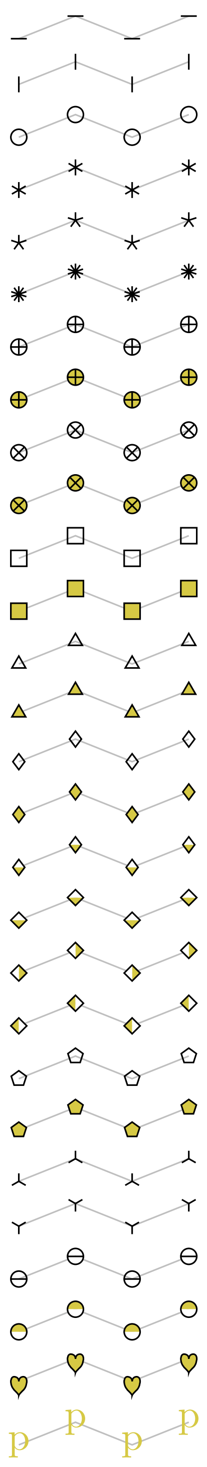
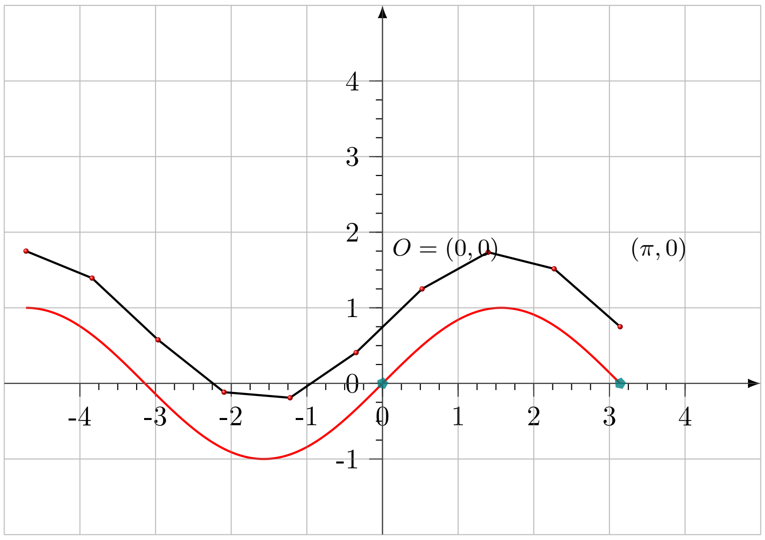


图 1.4: Point Marker

```
\begin{tikzpicture}[>=Latex]
    \xAxis[-5][5]
    \yAxis[-2][5]
    \Plot[-1.5*pi:pi][red, thick]{sin(x)}
    \ShowGrid[gray, step=1, opacity=.5]{(-5, -2); (5, 5)}
    % marker option
    \PlotPrecise{plot}{10}
    \Plot[-1.5*pi:pi][thick][type=ball, color=red]{sin(x)+.75}
    % show point
    \ShowPoint[color=teal, radius=2pt, type=pentagon*, opacity=.8, rotate=60]
        {(0, 0); (3.1415926, 0)}[$O=(0, 0)$; $(\pi, 0)$][above right=4em and 0em,
        font=\small]
\end{tikzpicture}
```



注意:zTikZ 中的命令都不需要你使用 “;” 去结束绘制.

1.3.7 Contour Plot

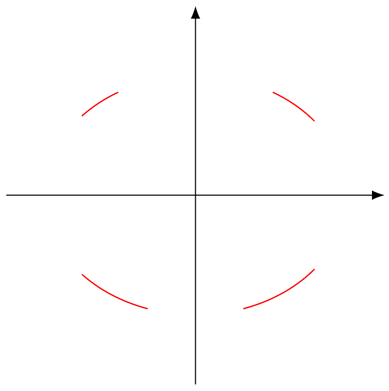
这几个函数绘制命令中，稍微值得一提的是命令:`\ContourPlot` ，其参数格式为:

```
\ContourPlot[<plot domain>][<plot style>][<marker options>]{<function>}
```

因为是 contour plot, 所以它的定义域指定格式比较特别; 比如绘制的区域为: $-3 < x < \pi$ 并且 $-1.5 < y < e$ 时. 那么在指定其 `<plot domain>` 时应写为 `-3:pi;-1.5:exp(1)`.

由于 zTikZ 的这部分功能都是以 gnuplot 为基础, 所以只要是 gnuplot 支持的函数, gnuplot 内置的任何常数；你都可以在 zTikZ 中使用；这里给不熟悉 gnuplot 的你们推荐一份 7 页的 gnuplot 快速入门清单:gnuplot card

这里给出一个关于\ContourPlot 命令的小例子:



```
\begin{tikzpicture}[>=Latex, scale=.5]
    \ShowAxis{(-5, 0); (5, 0)}
    \ShowAxis{(0, -5); (0, 5)}
    \ContourPlot[-3:pi; -3:exp(1)][red]{x**2/16 + y**2/10 - 1}
\end{tikzpicture}
```

对于\ContourPlot 还有一点提醒：若要绘制方程 $x^2/4 + y^2/9 = 1$，那么你只需输入 x**2/4+y**2/9-1，不需要对应的 $=$；所以由此也暗示了此命令的另一用法: 用于绘制水平线 ($y = c$) 和竖直线 ($x = c$). 仍然可以使用键 <plot domain> 控制变量 x, y 范围. 比如绘制 $x = 1, -1 < y < 1$. 那么对应的命令为:(第一个参数 x 的范围只要包含 $x = 1$ 即可):

```
\ContourPlot[0:2; -1:1][red, dashed]{x-1}
```

1.3.8 3d plot

目前调用 gnuplot 模块进行 3d 绘制的命令**仅有**\Plotz ，其格式为:

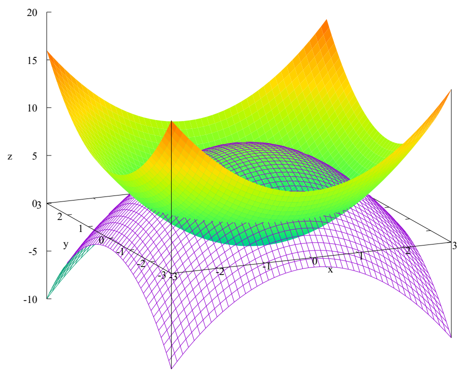
```
\Plotz[<domain>, <palette>, <width>]{<function>}
```

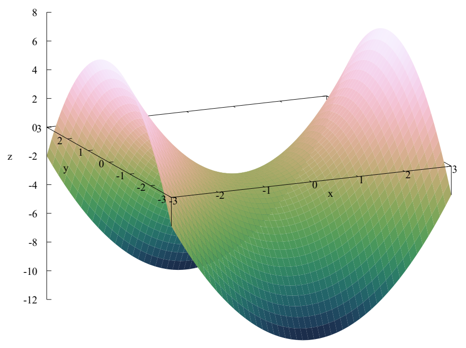
其中参数 <domain> 和命令\ContourPlot 中的格式和意义一致. 参数 <palette> 用于设置绘制的调色板，默认为 rgbformulae 22,13,-31，若想要自定义对应的调色板请参见 gnuplot 文档. 参数 <width> 接受一个合法的长度值，如.5\linewidth ，10cm，100pt 等.

下面为两个绘制示例:

```
\Plotz[
  width=1\linewidth,
  domain={-3:3; -3:3}
]{x**2+y**2-2 with pm3d, -x**2-y**2+8 with lines}
```

```
\Plotz[
  width=1\linewidth,
  domain={-3:3; -3:3},
  palette={cubehelix start 0 cycles -1. saturation 1}
]{x**2-y**2-2 with pm3d}
```





注记 1.3.5 关于绘制三维参数方程，目前不打算支持. 一方面是因为目前这个模块已经比较臃肿，另一方面则是 zTikZ 没有支持三维图形绘制的计划.

1.3.9 Intersection

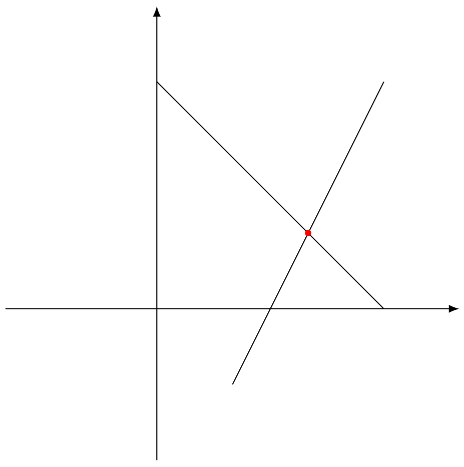
命令没有讲到:`\ShowIntersection` 的使用比较简单; 其命令`\ShowIntersection` 的调用格式为:

```
\ShowIntersection{<path 1>; <path 2>}[<number of points>]
```

指定 tikz 中 path 名称并显示交点方法示例, 我们分别指定两条叫做 `line1`, `line2` 的路径, 并显示它们二者的交点.

```
\begin{tikzpicture}[>=Latex]
    \ShowAxis{(-2, 0); (4, 0)}
    \ShowAxis{(0, -2); (0, 4)}
    \Plot[1:3][name path=line1]{2*x-3}
    \Plot[0:3][name path=line2]{-x+3}
    \ShowIntersection[color=red]{line1; line2}{1}

    % 可以使用如下的语句
    % \path[name intersections={of=line1 and line2}];
    % \ShowPoint[color=red] {(intersection-1)}
\end{tikzpicture}
```



1.3.10 PlotPrecise

`\PlotPrecise` 命令主要用于指定以上一个函数绘制命令的绘制精度, 其参数格式为:

```
\PlotPrecise{<plot type>}[<change domain>]{<samples-int>}
```

支持的 `<plot type>`:`plot`, `param`, `contour`, `polar`, 分别用于设置命令`\Plot` , `\ParamPlot` , `\ContourPlot` , `\PolarPlot` 的采样精度. 采样精度的设置分为两种, 临时和永久, 临时改变 (只改变下一个命令的采样精度) 的方法是在命令的第二个参数中填入 `[once]`; 而如果填入不是 `once`, 那么接下来的所有同种 `<plot type>` 的函数绘制命令的采样精度均会受到影响. 下面给出一个采样精度设置的例子, 绘制在区间 $[-2, 2]$ 上的函数 $y = 3\sin(1/x)$ 在采样精度分别为 50 和 1000 的图像, 见图 1.5:

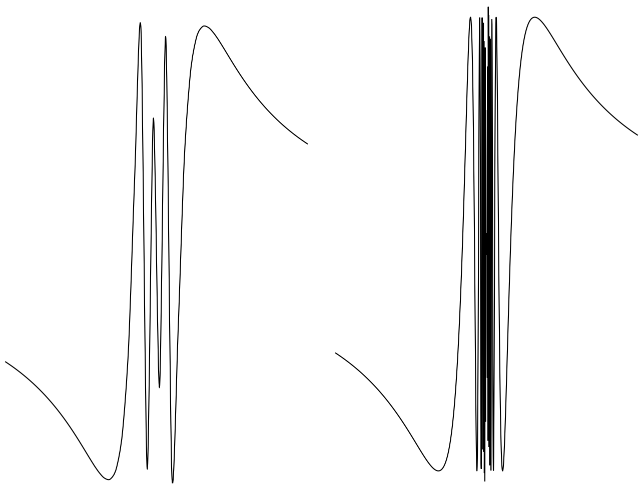


图 1.5: 绘制精度设置

1.4 Statistic Plot

1.4.1 Stairs Plot

了解完用户曲线绘制的一系列 Plot 命令后，现在开始介绍和统计相关的几个绘图命令: `\ListPlot` , `\StairsPlot` , `\StemPlot` , `\BarPlot` , `\ShadePlot` . 这几个命令在本节开篇已经介绍了其大致的功能. 本节主要聚焦于怎么使用这几个命令. 由于除`\ShadePlot` ,`\ListPlot` 外,其余 3 者的用法完全一致,不同的仅仅为第一个可选参数的意义. 其中`\ListPlot` 仅仅只需把对应的绘制命令中 `opacity=0` 即可，本节便以`\StairsPlot` 为例，讲解这三者的用法. 同时也会给出 3 者的具体示例.

`\StairsPlot` 主要用于绘制阶梯图, 命令格式为:

```
\StairsPlot[<stairs option>][<draw option>][<marker option>]{<data>}
```

第一个可选参数的可选值有:`plot-left(plot-right);jump-right(plot-left)`, 我们以 “;” 进行两个配置项的分割. 第一个配置项 `plot-<.>` 表示在绘制数据点时水平线取值为左值还是右值. 第二个配置项 `jump-<.>` 表示在绘制数据点时跳过左侧还是右侧的垂直线，为空时默认绘制两侧的垂直线.

注记 1.4.1 如果两个配置项 `plot-<.>`，`jump-<.>` 均为空，那么此时绘制的图像为通过线段相连的散点图

其中必选参数 `<data>` 表示要可视化的数据文件, 可以手动输入数据文件地址，也可以使用前文定义的`\gnudata {<index>}` 命令用于引用前面已经生成的数据, **这也就意味着要使用此命令，必须先产生对应的数据**. 一个绘制样例如图 1.6:

```
\begin{tikzpicture}
    \ShowGrid[step=1]{(-5, -5); (5, 5)}
    %% \StairsPlot
    % 1. connected using piecewise constant series of lines
    \begin{scope}[yshift=1cm]
        \StairsPlot[plot-left;][red]{./data/sine.data}
        \node[left] at (-3.14, 0) {plot left(default)};
    \end{scope}
    \begin{scope}[yshift=-1cm]
        \StairsPlot[plot-right;][orange]{./data/sine.data}
        \node[left] at (-3.14, 0) {plot right};
    \end{scope}
    % 2. plot segment (non-connected series of lines)
    \begin{scope}[yshift=2cm]
        \StairsPlot[;jump-left][teal][type=ball, color=teal]{./data/sine.data}
        \node[left] at (-3.14, 0) {jump left};
    \end{scope}
    \begin{scope}[yshift=-2cm]
        \StairsPlot[;jump-right][cyan][type=ball, color=cyan]{./data/sine.data}
        \node[left] at (-3.14, 0) {jump right};
    \end{scope}
\end{tikzpicture}
```

其余的两个相似命令`\StemPlot` , `\BarPlot` 的参数格式为:

```
\StairsPlot[<stem/bar option>][<draw option>][<marker option>]{<data>}
```

这里主要说明第一个可选参数的可选值列表, 默认值以及其对应的实际含义.`\StemPlot` 命令各参数含义如下:

- x: 表示绘制一个垂直于 x 轴上的火柴棍图
- y: 表示绘制一个垂直于 y 轴上的火柴棍图
- o: 表示绘制一个中心为原点 $O = (0, 0)$ 的火柴棍图
- 若为空值, 那么默认和 x 参数时的情形一致

其实命令 `\BarPlot` 和上述的`\Stemplot` 命令的使用方法完全类似, 一般人们把这个叫做 “火柴棒” 图. 此命令各参数及其含义如下:

- x: 表示绘制一个垂直于 x 轴上的**离散**柱形图
- y: 表示绘制一个垂直于 y 轴上的**离散**柱形图
- xc: 表示绘制一个垂直于 x 轴上的**连续**柱形图
- yc: 表示绘制一个垂直于 y 轴上的**连续**柱形图

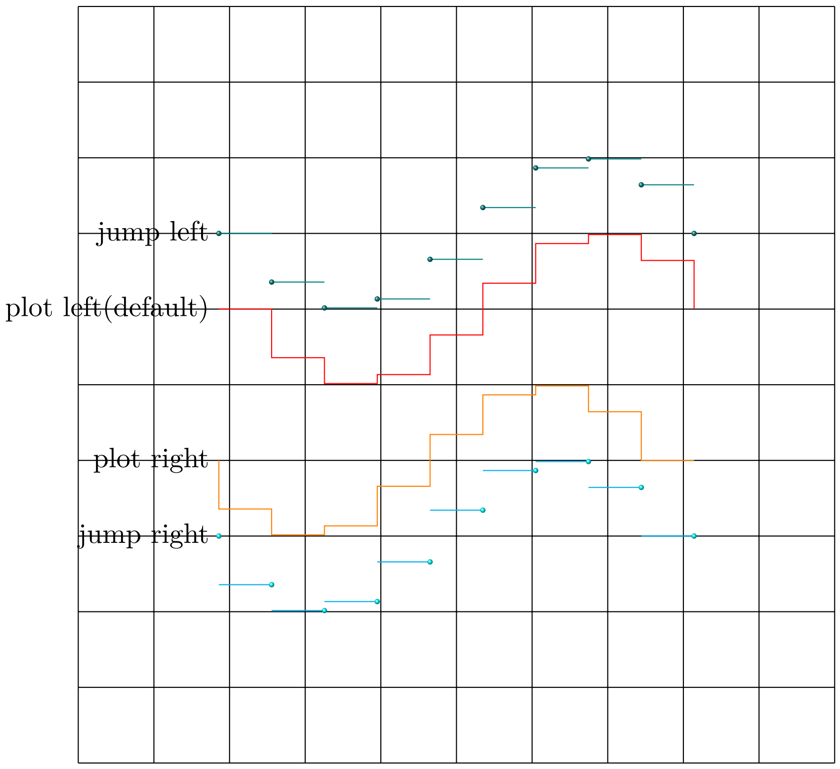


图 1.6: stairs-plot

- 若为空值，那么默认和 x 参数时的情形一致

1.4.2 Stem/Bar Plot

下面各处这两个命令的一个简单示例, 见图 1.8, 图 1.7:

```
\begin{tikzpicture}
    \ShowGrid[step=1]{(-5, -5); (5, 5)}
    %% \StemPlot
    % 1. xcomb
    \begin{scope}[yshift=-3cm]
        \StemPlot[x][red][type=*, color=red]{./data/sine.data}
        \node[left] at (-3.14, 0) {xcomb};
    \end{scope}
    % 2. ycomb
    \begin{scope}[yshift=0cm]
        \StemPlot[y][orange][type=*, color=orange]{./data/sine.data}
        \node[left] at (-3.14, 0) {ycomb};
    \end{scope}
    % 3. polar comb
    \begin{scope}[yshift=3cm]
        \StemPlot[o][cyan][type=*, color=cyan]{./data/sine.data}
        \node[left] at (-2, 0) {polar comb};
    \end{scope}
\end{tikzpicture}
```

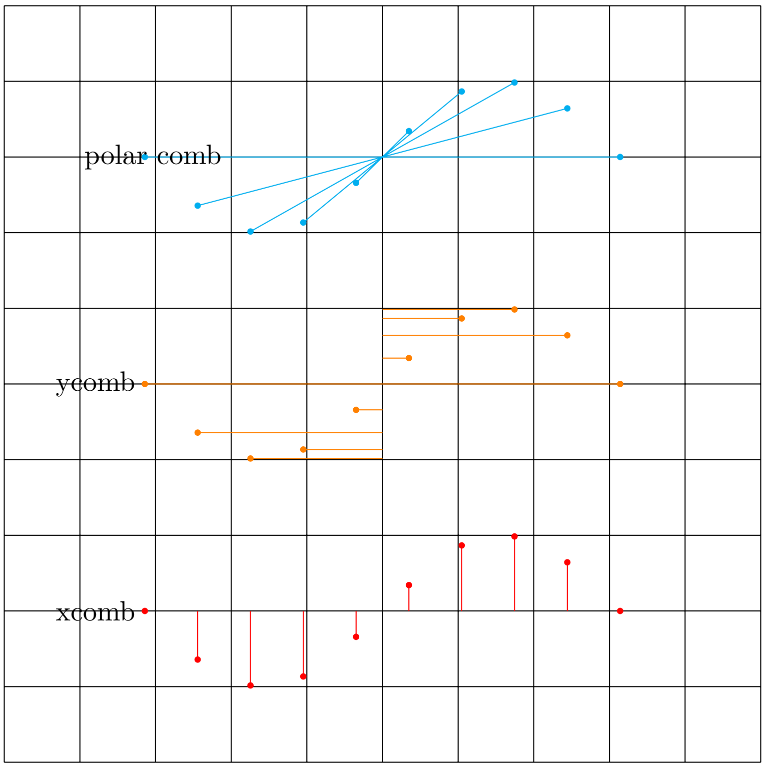


图 1.7: stem-plot

```
\begin{tikzpicture}
    \ShowGrid[step=1, color=gray, opacity=.5]{(-5, -5); (5, 5)}
    % 1. xbar
    \begin{scope}[yshift=-1cm]
        \BarPlot[x][red, pattern=north west lines, pattern color=red]{./data/sine.data}
        \node[left] at (-3.14, 0) {xbar};
    \end{scope}
    % 2. ybar
    \begin{scope}[yshift=1cm]
        \BarPlot[y][orange, pattern=north west lines, pattern
        color=orange]{./data/sine.data}
        \node[left] at (-3.14, 0) {ybar};
    \end{scope}
    % 3. xbar interval (fill)
    \begin{scope}[yshift=4cm]
        \BarPlot[xc][cyan, pattern=north west lines, pattern
        color=cyan]{./data/sine.data}
        \node[left] at (-3.14, 0) {xbar continuous};
    \end{scope}
    % 3. ybar interval (fill)
    \begin{scope}[yshift=-3cm]
        \BarPlot[yc][green, pattern=north west lines, pattern
        color=green]{./data/sine.data}
        \node[left] at (-3.14, 0) {ybar continuous};
    \end{scope}
    % annotate
    \node at (2.25 ,-4.5) {Optional args:bar width, bar shift};
\end{tikzpicture}
```

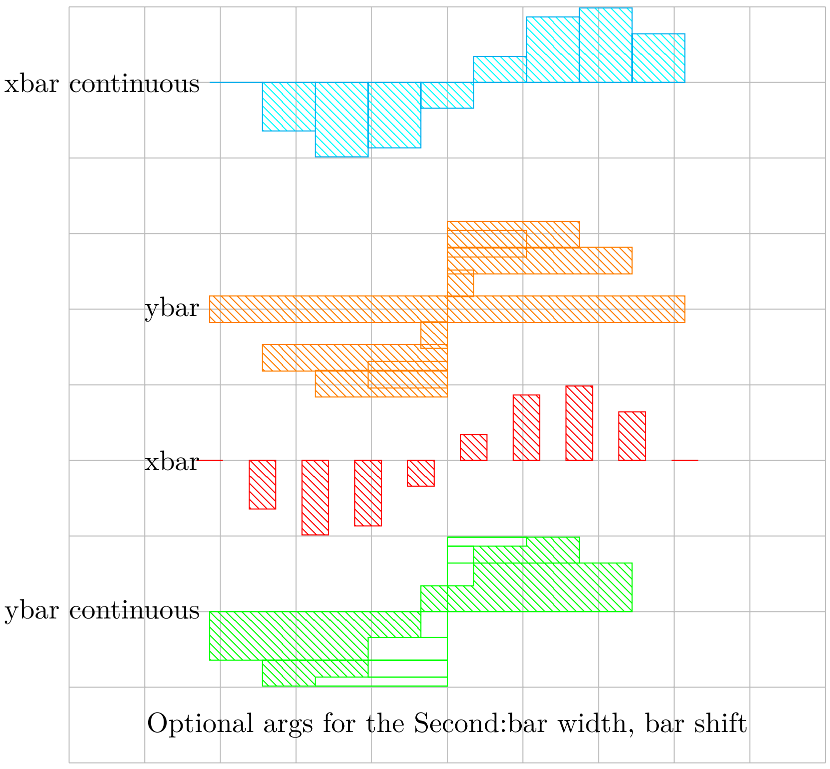


图 1.8: bar-plot

1.4.3 List Plot

然后讲解一下怎么绘制散点图，以\Plot ，\PolarPlot 命令为例，见图 1.9:

```
\begin{tikzpicture}
    \ShowGrid[step=1, color=gray, opacity=.5]{(-5, -5); (5, 5)}
    \begin{scope}[yshift=-2cm]
        \PlotPrecise{plot}{10}
        \Plot[-pi:pi][opacity=0, red][type=o, color=red]{sin(x)}
    \end{scope}
    \begin{scope}[yshift=3cm]
        \PlotPrecise{polar}{15}
        \PolarPlot[0:2*pi][opacity=0, orange][type=square, color=orange]{1-sin(t)}
    \end{scope}
    % continuous condition
    \Plot[-pi:pi][red]{sin(x)}
    \PolarPlot[0:2*pi][orange]{1-sin(t)}
\end{tikzpicture}
```

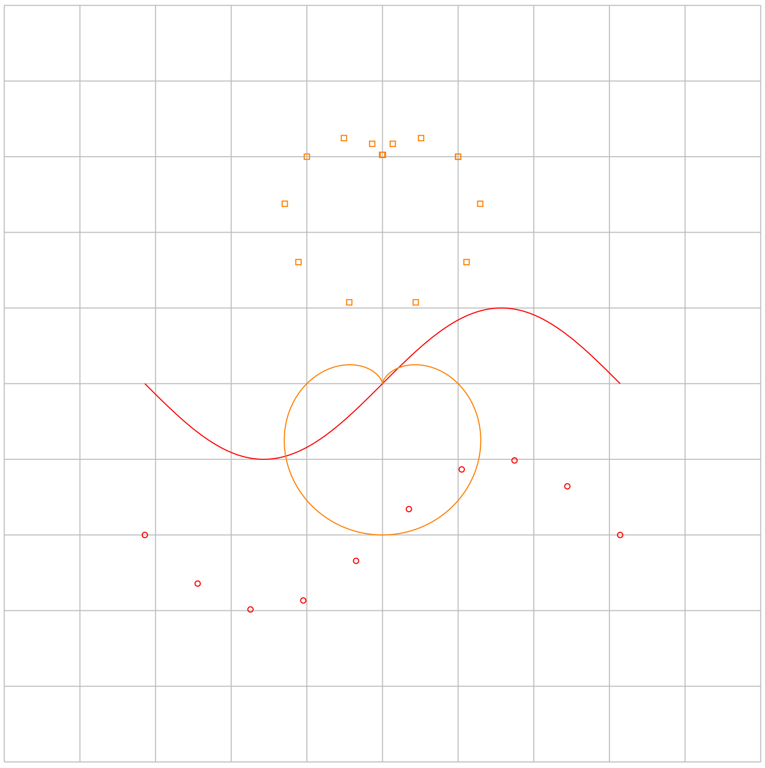


图 1.9: list-plot

1.5 zdraw

1.5.1 基本命令

zTikZ 目前包含一基于 l3draw 的子模块 zdraw, 用于满足一些比较简单的绘图需求.zdraw 模块提供了两个基本的绘制命令和一个基本环境.

- `\zrule` : 用于绘制一个支持颜色渐变的 rule(矩形)
- `\zplot` : 绘制支持颜色渐变的形如 $y = f(x)$ 的普通函数

zdraw 模块提供的 `\zrule` 的命令使用是比较简单的, 格式如下:

```
\zrule[<width>, <height>, <startColor>, <endColor>, <step>]
```

上述参数中的 `<step>` 表示绘制渐变时的精度，自然对应的数值越小，那么对应的色彩过渡越自然. 比如，用户可以使用这个命令来定义 `\headrule` ，从而生成一个颜色渐变的 fancyhead rule. 具体代码如下:

```
\pagestyle{fancy}
\setlength{\headheight}{25pt}
\def\headrule{\zdrawSetUnit{pt}\zrule[width=\textwidth, step=1, height=1]}
```

下面例举一个`\zrule` 命令的使用示例，以此来说明此命令中各个参数的含义:

```
\zrule[width=10, startColor=red, step=1]
```

关于命令 `\zplot` 的使用细节请参见:小节 1.5.4. 这里介绍如何设置 `\zplot` 绘图时的单位以及绘制的线径参数. zdraw 提供 `\zdrawSetUnit` , `\zdrawSetPathWidth` 用于设置坐标单位和线径. 一个基本的使用样例见下:

```
\zdrawSetUnit{cm}
\zdrawSetPathWidth {5pt}
```

l3draw 中默认的线径为 `0.4pt`. 那么怎么设置 l3draw 中默认的单位坐标尺寸呢? 参见如下代码:

```
\draw_xvec:n {2cm, 0}
\draw_yvec:n {0, 2cm}
...
\draw_path_lineto:n {\draw_point_vec:nn {<x-coor>}{<y-coor>}}
...
```

上述的 `x-coor`，`y-coor` 即代表纯数字的坐标值. 通过修改上述代码中 x-axis 方向的基向量`\draw_xvec:n{2cm,0}` 或 y-axis 方向的基向量，用户即可以自定义绘图的单位坐标尺寸.

1.5.2 基本环境

zdraw 同时也提供了一个基本的 `zdraw` 环境，用于用户自由书写对应的绘图代码. 在提供本环境的同时 zdraw 模块对原 l3draw 宏包中的部分命令以及部分 LaTeX3 中的命令进行了封装. 源码声明如下:

```
%% draw env
\NewDocumentEnvironment{zdraw}{O{}}{\ExplSyntaxOn\draw_begin:}{\draw_end:\ExplSyntaxOff}
\NewDocumentEnvironment{zscope}{O{}}{\draw_path_scope_begin:}{\draw_path_scope_end:}

%% cmd alias
% point/line
\cs_new_eq:NN \moveto \draw_path_moveto:n
\cs_new_eq:NN \lineto \draw_path_lineto:n
\cs_new_eq:NN \scolor \color_stroke:n
\cs_new_eq:NN \fcolor \color_fill:n

% base vector
\cs_new_eq:NN \xvec   \draw_xvec:n
\cs_new_eq:NN \yvec   \draw_yvec:n
% both follows ref (0, 0)
\cs_new_eq:NN \polar  \draw_point_polar:nn
\cs_new_eq:NN \coor   \draw_point_vec:nn

% shape and text
\cs_new_eq:NN \rec \draw_path_rectangle_corners:nn
\cs_new_eq:NN \cir \draw_path_circle:nn
\cs_new_eq:NN \newtext    \coffin_new:N
\cs_new_eq:NN \sethtext   \hcoffin_set:Nn
\cs_new_eq:NN \setvtext   \vcoffin_set:Nnn
\cs_new_eq:NN \scaletext  \coffin_scale:Nnn
\cs_new_eq:NN \puttext    \draw_coffin_use:Nnnn

% path operation
% scope
\cs_new_eq:NN \bg \draw_path_scope_begin:
\cs_new_eq:NN \eg \draw_path_scope_end:
% segement connect
\cs_new_eq:NN \capbutt   \draw_cap_butt:
\cs_new_eq:NN \caproun   \draw_cap_round:
\cs_new_eq:NN \caprect   \draw_cap_rectangle:
\cs_new_eq:NN \closepath \draw_path_close:
% transformatio
\cs_new_eq:NN \shift     \draw_transform_shift:n
\cs_new_eq:NN \xscale    \draw_transform_xscale:n
\cs_new_eq:NN \yscale    \draw_transform_yscale:n
\cs_new_eq:NN \trans     \draw_transform_matrix:nnnn
\NewDocumentCommand{\usepath}{O{draw}}{
  \draw_path_use_clear:n {#1}
}
```

上述的命令:\trans {a}{b}{c}{d} 需要稍微说明, 此命令表示对整个坐标系统进行一个线性变换, 其所对应的变换矩阵 T 如下:

$$T = \begin{bmatrix} a & c \\ b & d \end{bmatrix}$$

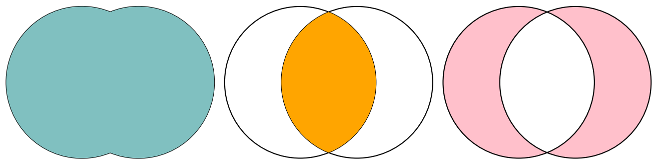
其余的命令请参见具体文档, 这里以一个基本的例子–绘制 Venn 图, 来讲解上述命令的基本使用方法.

```
% union
\begin{zdraw}
  \xscale {0.5}
  \yscale {0.5}
  \cir {2cm, 0}{2cm}
  \cir {3.5cm, 0}{2cm}
  \usepath[draw, clip]
  \fcolor {teal!50}
  \rec {-10cm, -10cm}{10cm, 10cm}
  \usepath[fill]
\end{zdraw}

% intersection
\begin{zdraw}
  \xscale {0.5}
  \yscale {0.5}
  \cir {3.5cm, 0}{2cm}
  \usepath[draw]
  \cir {2cm, 0}{2cm}
  \usepath[clip, draw]
  \fcolor {orange}
  \cir {3.5cm, 0}{2cm}
  \usepath[fill]
\end{zdraw}

% difference
\begin{zdraw}
  \xscale {0.5}
  \yscale {0.5}
  \draw_evenodd_rule:
  \fcolor {pink}
  \cir {2cm, 0}{2cm}
  \cir {3.5cm, 0}{2cm}
  \usepath[draw, fill]
\end{zdraw}
```

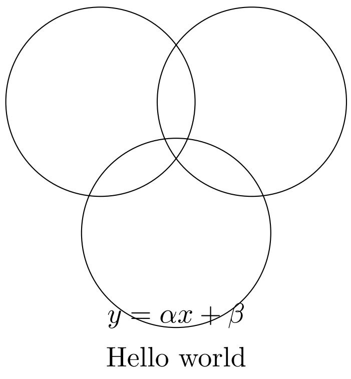
绘制效果如下:



目前，在 l3draw 中插入文本是通过插入一个 coffins 或者是 box 实现的，下面显示一个插入文本的建档示例:

```
\begin{zdraw}
  \xscale {.5}
  \yscale {.5}
  \cir {-2cm, 0}{2.5cm}
  \cir {2cm, 0}{2.5cm}
  \cir {0, -2*sqrt(3)cm}{2.5cm}
  % text
  \newtext \texta
  % \sethtext \texta {$y=\alpha x + \beta$\\ Hello~ world}
  \setvtext \texta {6em}{$y=\alpha x + \beta$\\ Hello~ world}
  \scaletext \texta {2}{2}
  \puttext \texta {hc}{b}{0, -7cm}
  \usepath[draw]
\end{zdraw}
```

具体效果如下:



注记 1.5.1 目前这些命令的设置是不太合理的，也仅为了我个人使用方便. 所以如果用户想要深入了解 l3draw，那么还请自行阅读其对应的文档，而非使用 zdraw 模块中 “封装”(alias) 的如上命令.

1.5.3 pgf Shade Plot (deprecated)

zTikZ 之前的 gnuplot 模块默认提供了一个基于 pgf 的`\ShadePlot` 命令，参数格式为:

```
\ShadePlot[<shade mode>][<box distance>]{<data>}
```

其中第一个参数 `<shade mode>` 使用命令`\ztikzShadeMode` 命令进行声明，此命令的一个样例为 (zTikZ 默认 shade 模式):

```
\ztikzShadeMode{defaultMode}{horizontal}{white,black}
```

`\ztikzShadeMode` 的第二个参数可以为 `vertical`，表示垂直渐变. 第三个参数中的颜色也可以超过两个. 一个具体的使用样例, 见图 1.10:

```
% compile with "\usepackage[external=false]{ztikz}"
\ztikzShadeMode{newyMode}{vertical}{white,black}
\begin{tikzpicture}
    \ShowGrid[step=1, color=gray, opacity=.5]{(-5, -5); (5, 5)}
    \ShadePlot[newyMode][10pt]{./data/sine.data}
\end{tikzpicture}
```

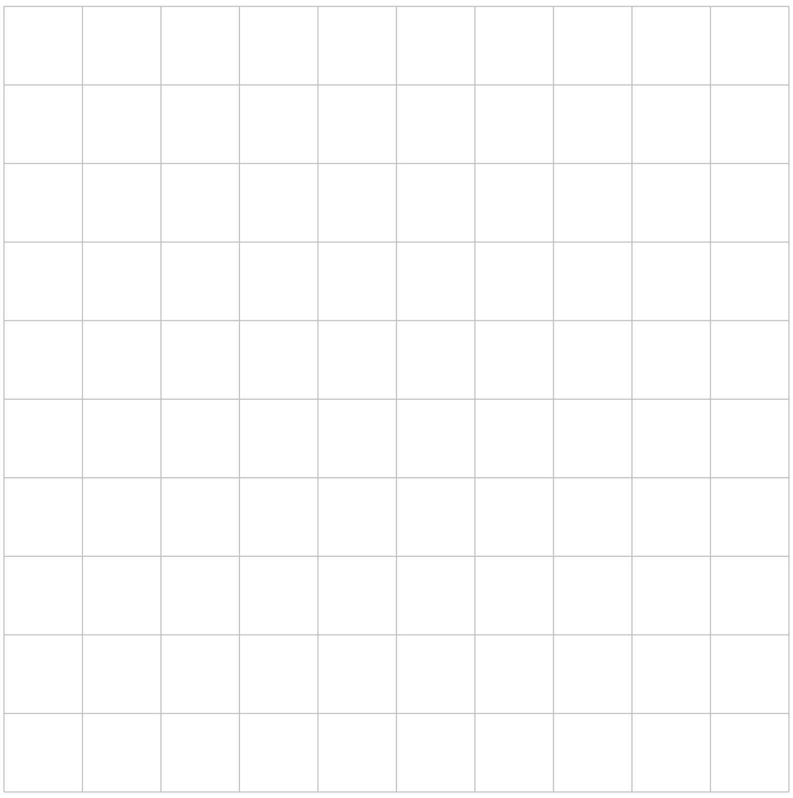


图 1.10: shade-plot

目前此命令可能会存在部分的潜在问题, 请谨慎使用此命令 (比如此命令不支持 `external` 库的缓存功能). 如果想要创建一个 shade 区域，直接使用 `\shade` 命令来代替`\fill` 命令即

可. 在指定`\shade [<shade mode>] ...` 命令中的 `<shade mode>` 时可以为:`top color=red, bottom color=blue` 或者是 `left color=red, right color=blue`.

注记 1.5.2 目前基于 pgf 的 `\shadeplot` 命令已经被废弃，因为其在绘制时会出现一些兼容性问题，但是在后续的 implement 节我仍然把之前的实现细节给出了. 如果使用者有类似的需求, 那么请使用模块 zdraw 中此命令的新的实现方法`\zplot` ，而非此命令原始实现. 但是其实 zdraw 模块中的 shade 模式也并不是正真意义上的 gradient shading, 而是通过多段 gradient segements 拼接而成的. 什么时候 zdraw 能支持真正意义上的 gradient? 这取决于 l3draw 官方是否考虑加入此功能，毕竟这个功能需要一些 PDF management 的东西. 详情请参见: smooth gradient in l3draw.

1.5.4 l3daw Shade Plot

由于上述基于 pgf 的`\shadeplot` 命令的种种局限性. 目前 zdraw 模块已通过 l3draw 模块实现了曲线的渐变填充 (并不是在原理层面解决的，l3draw(LaTeX3)) 官方正在考虑是否加入此功能). 但是由于 l3draw 模块的限制，目前在 l3draw 模块中实现的命令还不支持在 tikz 环境中使用. zdraw 模块中提供的命令 `\zplot` 的格式为:

```
\zplot[<domain>, <range>, <action>, <startColor>, <endColor>, <axis>]{<function>}
```

上述的其余参数均与命令 `\zrule` 中参数的意义一致, 需要介绍的重点参数为 `<range>`. 此参数表示所绘制函数的范围. 这个参数在指定颜色的渐变范围时是比较有用的.

注记 1.5.3 但是总的说来: 这个颜色渐变这个功能你其实并不是经常使用的

在 l3draw 中实现的命令`\function_plot:nnn` 可以完成水平方向和垂直方向的渐变, 一个具体的示例如下

```
\zplot[domain={0, 0.02*\c_pi_fp, 2*\c_pi_fp}, action=shade, startColor=blue,
endColor=green, axis=x]{sin(x)}
\zplot[domain={0, 0.02*\c_pi_fp, 2*\c_pi_fp}, action=shade, startColor=blue,
endColor=green, axis=y]{sin(x)}
```

具体绘制效果见图 1.11:

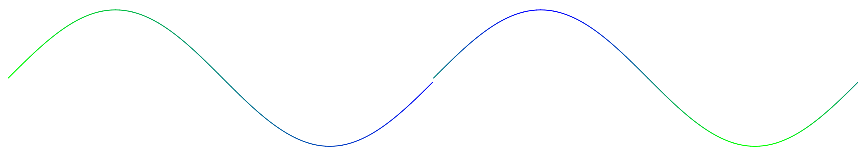


图 1.11: l3draw gradient

关于此命令的具体实现，请参见节 4.4. 当然，你也可以指定 `startColor,endColor` 为同一个颜色，从而绘制普通的函数曲线. 如果要实现更多的渐变方式，比如沿着曲线渐变，可以参考如下示例:

```
% \usepackage{l3draw}
\ExplSyntaxOn
\seq_set_split:Nnn \l_tmpa_seq {;}{
  0.00000, 0.00000;
  0.03015, 0.00990;
  -0.04733, -2.98255;
  ...
  0.89511, -2.87871;
  1.76708, -2.48152;
  2.47872, -1.82505;
  2.95459, -0.97038;
  3.14159, -0.00000
}
% color gradient
\cs_set:Npn \color_gradient:n #1 {
  \color_select:n {blue!#1!green}
}
\cs_generate_variant:Nn \seq_set_split:Nnn {Nnx}
\cs_generate_variant:Nn \draw_path_moveto:n {x}
\cs_generate_variant:Nn \color_gradient:n {x}

\draw_begin:
\draw_cap_round:
% base
\draw_xvec:n {2cm, 0}
\draw_yvec:n {0, 2cm}
% draw
\draw_path_moveto:n {\draw_point_vec:nn {0.785}{0}}
\int_step_inline:nnn {2}{\fp_eval:n {\seq_count:N \l_tmpa_seq-1}}{
  \seq_set_split:Nnx \l_tmpb_seq {,}{\seq_item:Nn \l_tmpa_seq {#1}}
  \seq_set_split:Nnx \l_tmpc_seq {,}{\seq_item:Nn \l_tmpa_seq {\fp_eval:n {#1+1}}}
  \color_gradient:x {\fp_eval:n {#1*100/\seq_count:N \l_tmpa_seq}}
  \draw_path_moveto:n {
    \draw_point_vec:nn {\seq_item:Nn \l_tmpb_seq {1}}{\seq_item:Nn \l_tmpb_seq {2}}
  }
  \draw_path_lineto:n {
    \draw_point_vec:nn {\seq_item:Nn \l_tmpc_seq {1}}{\seq_item:Nn \l_tmpc_seq {2}}
  }
  \draw_path_use_clear:n {draw}
}
\draw_path_use_clear:n {draw}
\draw_end:
\ExplSyntaxOff
```

具体效果为:

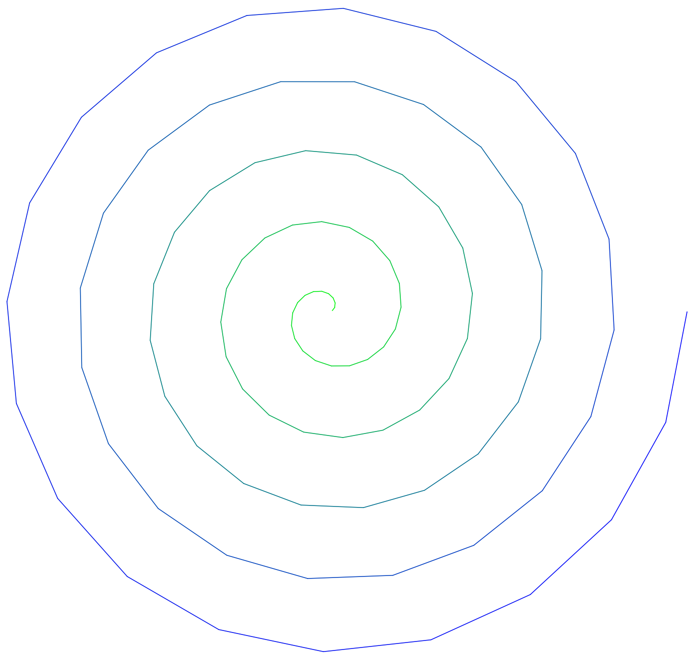


图 1.12: Curve Gradient

1.6 Polygon

1.6.1 基本介绍

目前 zTikZ 的 Polygon 命令已经开发完毕, 此命令主要用于绘制正多边形. 命令格式如下:

```
\Polygon[<options>]{<edges>}
```

第二个参数是比较简单的，只需要输入一个正整数即可，表示多边形的边数. 第一个参数是一个可选参数，可以指定旋转角度, 多边形的边长, 边的颜色多边形填充样式与颜色，顶点样式等. 可用的键值对以及默认值如下:

```
\keys_define:nn { ztikz / polygon }{
    radius      .fp_set:N  = \l__polygon_radius_fp,
    radius      .initial:n = { 1 },
    edgeColor   .tl_set:N  = \l__polygon_edge_color_tl,
    edgeColor   .initial:n = { black },
    fillColor   .tl_set:N  = \l__polygon_fill_color_tl,
    fillColor   .initial:n = {  },
    fillOpacity .fp_set:N  = \l__polygon_fill_opacity_fp,
    fillOpacity .initial:n = { 0 },
    rotate      .fp_set:N  = \l__polygon_rotate_angle,
    rotate      .initial:n = { 0 },
    shift         .tl_set:N  = \l__polygon_shift_tl,
    shift         .initial:n = { (0,0) },
    marker        .tl_set:N  = \l__polygon_marker_option_tl,
    marker        .initial:n = { },
}
```

这里指的说一下的参数是:<marker>, 此参数中可以填入任意合法的 <marker option> 键值对, 一下为一个简单示例:

```
\Polygon [
    radius=2,
    shift={(3,0)},
    rotate=60,
    marker={type=ball, color=red},
    edgeColor=teal,
    fillColor=red,
]{3}
```

注记 1.6.1 多边形的变数必须为整数. 同时再说明一下键 <shift> 的含义: 合法的值为一个使用 "()" 包括的向量, 表示整个多边形的平移量. 比如 shift={(3,-4)} 表示整个多边形向右平移 3 个单位, 象下平移 4 个单位. 不要忘记向量外层的大括号:{}.

注记 1.6.2 其实也可以考虑把这个多边形的边数拓展为分数, 甚至是负数.

1.6.2 使用样例

在讲述了\Polygon 命令的用法后, 下面给出一个简单的使用样例, 见图 1.13:

```
\begin{tikzpicture}
    \ShowGrid[step=1, color=gray, opacity=.5]{(-5, -5); (5, 5)}
    \Polygon[shift={(0, 0)}]{3}
    \Polygon[shift={(3, 0)}, fillColor=red, fillOpacity=.5]{4}
    \Polygon[shift={(0, 3)}, edgeColor=green, fillOpacity=.3, marker={type=ball,
    color=green}, rotate=18]{5}
    \Polygon[shift={(-3,0)}, fillColor=orange]{6}
    \Polygon[shift={(0,-3)}, fillColor=teal, marker={type=oplus*, color=teal}]{8}
\end{tikzpicture}
```

注记 1.6.3 填充多边形时, 若没有指定 fillColor 时, 则不填充色彩. 当指定 fillColor 时, 默认填充透明度为 0.75. 用户可以重新指定其透明度值, Override It. 比如, 你想要填充一个透明度为 50% 的多边形, 那么可以使用 fillOpacity=0.5 来实现.

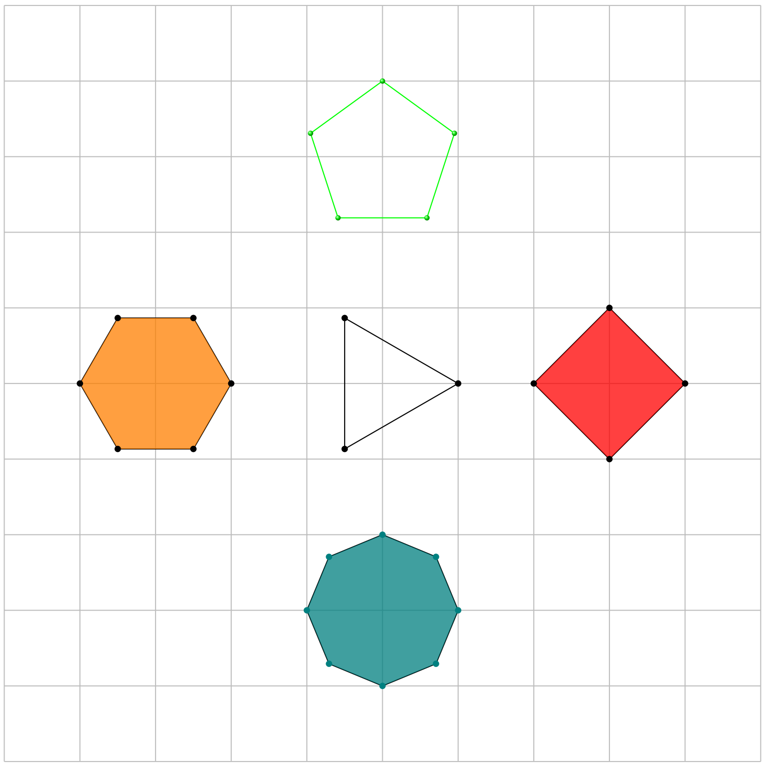


图 1.13: polygon

1.7 pyfig

1.7.1 基本介绍

python 绘图是比较就简单的，zTikZ 提供了用于 python 绘图的 pyfig环境. 此环境需要填入两个参数，参数格式为:

```
\begin{pyfig}[<width>]{<export file name>}
% your code
\end{pyfig}
```

其中的 <width> 参数是命令\includegraphics[<width>]{} 中的 width 参数, 比如你可以输入 width=.75\linewidth . 在指定必要的参数后，便可以直接在环境中输入 Python 代码. 下面即为一个示例:

```
\begin{pyfig}[width=.45\linewidth]{pycode.py}
import matplotlib
matplotlib.use('Agg')
from matplotlib import pyplot as plt
plt.rcParams['font.sans-serif'] = ['FangSong']
plt.rcParams['axes.unicode_minus'] = False
```

```
import numpy as np

x = np.linspace(0, 2*np.pi, num = 80)
y = np.sin(x)*np.cos(x)+.2
plt.plot(x, y)
\end{pyfig}
```

你不需要在源代码中输入图片的保存指令:`plt.savefig("")`, zTikZ 会自动在此环境后面加上对应的图片保存指令. 这个环境的返回结果为:`\includegraphics[width=.45\linewidth]{pycode.py.pdf`
所以你可以把这个环境嵌套在任何的浮动环境，比如 `figure`，`table` 中.

1.7.2 缓存机制

下面说一说这个 pyfig 环境的缓存机制. 在命令行中第一次编译此示例时你会看到如下的日志输出:

```
current hash is FF7B5ECDBF52AA95DF921FCC076F9021
current hash is unique --> recorded
```

上述日志说明，zTikZ 已经识别到这是一个新的 python 环境，并且保存了这个环境中绘图代码的 Hash 值；然后，第二次编译此文档时，你会在输出的日志中定位到如下的输出:

```
current hash is FF7B5ECDBF52AA95DF921FCC076F9021
skip recompile by python, using the cache picture 1
```

这就说明，由于你的 python 绘图部分的源代码没有改变，然后 zTikZ 就直接采用了上一次编译的缓存图片, 跳过了重新编译这一步; 上面环境的运行结果参见:图 1.14

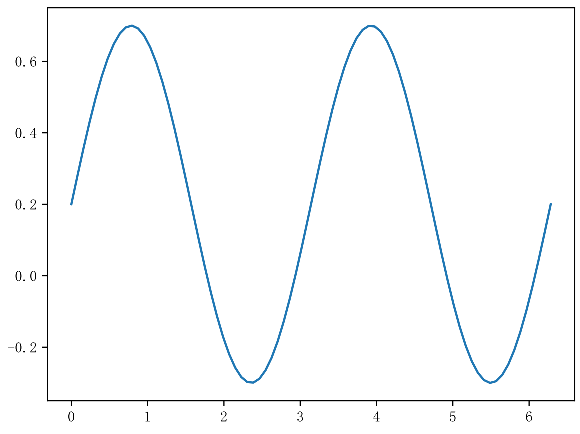


图 1.14: Python 绘图示例 1

1.7.3 3D 绘图示例

这里再给一个 Python 绘图示例, 绘制了一个来自 Matplotlib 官方的简单三维图像. 这里给出这个例子，就是为了让用户明白，尽管 zTikZ 没有提供便捷的用户绘制三维矢量图的接口，但是你可以使用 Python 生成对应的 3 维矢量图；虽然，你可能需要再去学习 Python 中 Matplotlib 的相关语法，但是这是简单的.

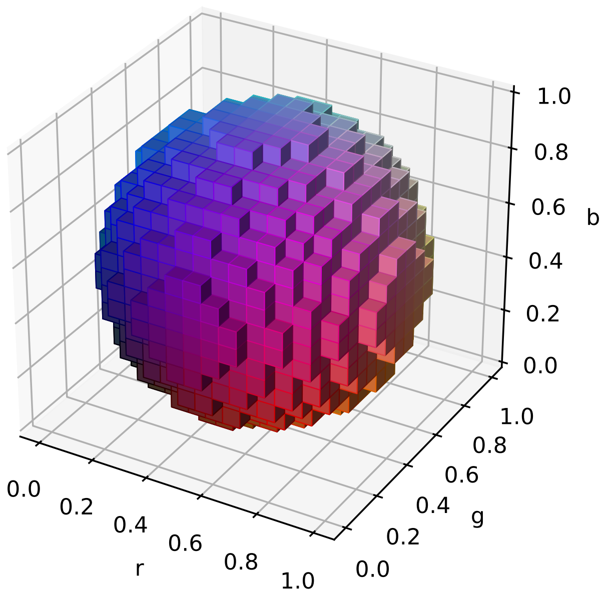


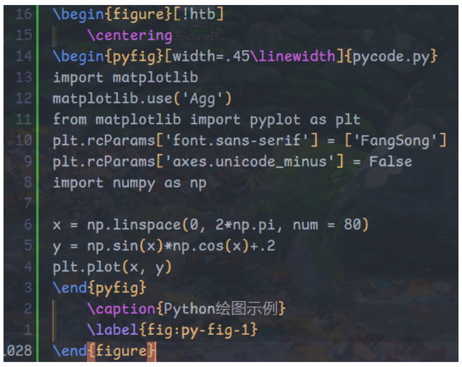
图 1.15: Python 绘图示例 2

1.7.4 注意事项

由于 python 是依靠缩进来识别代码结构的，所以在书写这部分的代码时，不能够人工添加缩进，在书写的时候需写为图 1.16这样.

如果你实在是需要缩进，那么在这里我推荐另外一种可以使用缩进的方法：把 `pyfig` 环境连同其内部代码保存在另外一个文件中，比如这里我保存为 `pycode.mpl`, 然后在 `figure` 环境中使用`\input{pycode.mpl}` 引入此部分的代码, 示例如下:

```
\begin{figure}
  \input{./data/pycode.mpl}
\end{figure}
```



```
\begin{figure}[!htb]
    \centering
\begin{pyfig}[width=.45\linewidth]{pycode.py}
import matplotlib
matplotlib.use('Agg')
from matplotlib import pyplot as plt
plt.rcParams['font.sans-serif'] = ['FangSong']
plt.rcParams['axes.unicode_minus'] = False
import numpy as np

x = np.linspace(0, 2*np.pi, num = 80)
y = np.sin(x)*np.cos(x)+.2
plt.plot(x, y)
\end{pyfig}
    \caption{Python绘图示例}
    \label{fig:py-fig-1}
\end{figure}
```

图 1.16: Python Indent

1.8 wolframGraphics

1.8.1 基本介绍

其实使用 mathematica 进行绘图这个部分和前面的使用 Python 绘图是差不多的，zTikZ 提供了一个 `wolframGraphics` 环境用于调用 mathematica 进行绘图. 与之前 `pyfig` 环境不同的是: 此时你需加入保存图片的命令, 路径前缀为:`./ztikz_output/mma_data/<figure name>`. 为何这里我不使用 zTikZ 自动完成? 由于 mathematica 绘图代码中可能存在着多幅图形需要互相组合的情况，可能会使用 `Show` 命令组合成为最终的目标图形, 但是这个组合方式就多种多样了.

所以为给用户提供更多自由空间,这里的图片保存命令由用户自己书写. 并且上述 `<figure name>` 只能写为 `<wls script name>.pdf` 的形式;比如你的 WolframScript 脚本名称为 `mma_1.wls`, 那么你的 `<figure name>` 只能写为 `mma_1.wls.pdf`, 其中的图片格式可以自己指定，可以为:`.png`，`.jpg`，`.mbp` 等. 此环境同样支持 cache 模块.

1.8.2 环境测试

用户如果要使用 zTikZ 的 Mathematica 模块，请务必确保 wolframscript 在命令行中能够正常运行. 可以使用如下文件作为测试用例，检测 wolframscript 是否正常工作;

```
plotFunction[fun_, xlimits_, ylimits_] := ContourPlot[fun,
    xlimits, ylimits,
    ContourStyle->{
        RGBColor["#00C0A3"],
        Thickness[0.004]
    },
    AspectRatio->((xlimits[[2]]//Abs) + (xlimits[[3]]//Abs))/((ylimits[[2]]//Abs) +
    (ylimits[[3]]//Abs)),
    AxesOrigin->{0,0},
    Axes->True,
    Frame->False,
    AxesStyle->Arrowheads[{0, 0.03}],
    AxesLabel->{"x", "y"},
    PlotRange -> Full
]

xlimits = {x, -3, 6};
ylimits = {y, -4, 5};
fp1 = plotFunction[y==Sin[x], xlimits, ylimits];
fp2 = plotFunction[x^2/4 + y^2/3 == 5, {x, -5, 5}, {y, -5, 5}];

figure = Show[fp2, fp1];
(* 1. 保存的图片格式为:*.wls.pdf; 2. 保存路径在:./ztikz_output/mma_data *)
Export["works_well.pdf", figure];
```

把这里的源码保存为 `test.wls`, 然后在命令行运行:

```
wolframscript -script test.wls
```

如果正常工作的话, 那么在你的当前工作目录下会产生一个名为 `works_well.pdf` 的 pdf 文件. 反之，你的 wolframscript 没有正常配置或者是激活, 也就不能够使用本模块.

1.8.3 2D 绘图示例

在介绍了 wolfram 模块的绘图功能后，下面给出一个具体的使用案例:

```
\begin{wolframGraphics}[width=.4\linewidth]{mma_1.wls}
    plotFunction[fun_, xlimits_, ylimits_] := ContourPlot[fun,
        xlimits, ylimits,
        ContourStyle->{
            RGBColor["#00C0A3"],
            Thickness[0.004]
        },
        AspectRatio->((xlimits[[2]]//Abs) + (xlimits[[3]]//Abs))/((ylimits[[2]]//Abs) +
        (ylimits[[3]]//Abs)),
        AxesOrigin->{0,0},
        Axes->True,
        Frame->False,
        AxesStyle->Arrowheads[{0, 0.03}],
        AxesLabel->{"x", "y"},
        PlotRange -> Full
    ]

    xlimits = {x, -3, 6};
    ylimits = {y, -4, 5};
    fp1 = plotFunction[y==Sin[x], xlimits, ylimits];
    fp2 = plotFunction[x^2/4 + y^2/3 == 5, {x, -5, 5}, {y, -5, 5}];

    figure = Show[fp2, fp1];
    (* 1. 保存的图片格式为:*.wls.pdf; 2. 保存路径在:./ztikz_output/mma_data *)
    Export["./ztikz_output/mma_data/mma_1.wls.pdf", figure];
\end{wolframGraphics}
```

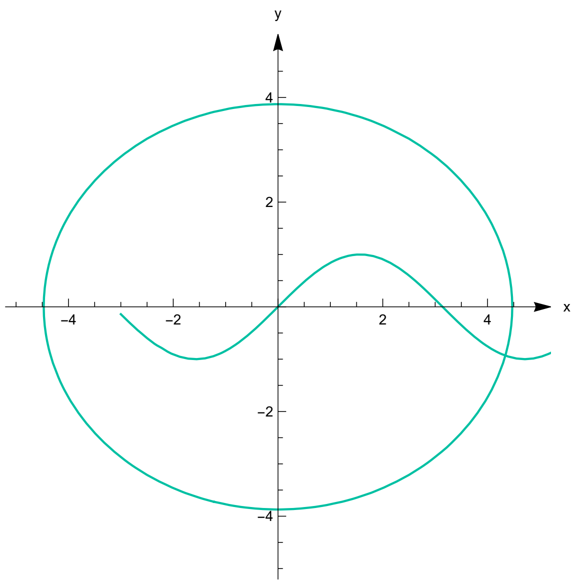


图 1.17: Mathematica 绘图示例

1.8.4 3D 绘图示例

同样的你可以使用 Mathematica 绘制 3 维图形[6]. 目前 zTikZ 仅支持插入静态图片，后续可能会考虑加入动态图片的支持功能，就像另外一个开源矢量图象绘制软件Asymptote 中的.prc 文件一样. 但是要使得能在 PDF 中预览动态图形，首先你的 PDF 阅读器必须支持 JavaScript，常见的这种类型的 PDF 阅读器就是 Adobe 家的 Acrobat 了.

1.8.5 注意事项

因为 mathematica 中的代码是允许用户自由添加缩进的，所以你可以按照个人喜欢的格式对 Mathematica 代码进行缩进的添加. 和前面的 Python 绘图代码类似，你可以把此部分代码保存在一个单独的文件中，然后通过\input 进行引入, 这里不再给出对应的案例. 有如下的几个注意事项:

➠ **空格与 Tab**

[6]由于目前的 Mathematica 不支持输出 3 维矢量图，所以想要是你的 3 维图像显得更加的清晰，可以调节图像的分辨率.

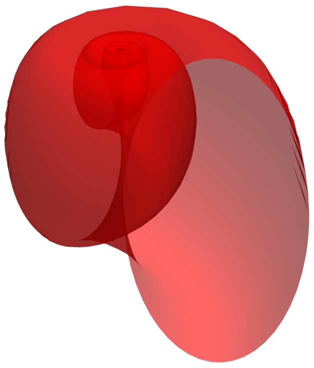


图 1.18: Mathematica 绘图示例 2

注意空格与 Tab，如果源代码中有 Tab，那么 zTikZ 在进行此环境的抄录时会把原本的 Tab 转义为 `^^I`, 从而造成 Mathematica 源代码的错误, 比如你可能会看到你的源代码抄录后变成了下面的样子:

```
^^IContourStyle->{
^^I^^IRGBColor["#00C0A3"],
^^I},
```

➠ 注释规范

同时注意 Mathematica 中注释的写法, 不是`(* something*)`, 而是`(* something *)`, 也就是你的注释不能够紧挨着 `*`, 否则会造成 mathematica script 的解析错误.

➠ 脚本名称

由于 WolframScript 的限制，对应的 Mathematica 脚本的后缀只能为:`.wls`，否则 WolframScript 无法识别此脚本，也就不会去执行此脚本了.

1.9 matlab

1.9.1 基本介绍

目前 zTikZ 中的 Matlab 模块还没有开发, 未来可能也没有这个打算了. 但是你可以使用插件: matlab2tikz. 这个 Matlab 插件可以把你的 Matlab 图形转换为对应的 tikz 代码, 经过测试发现, 效果也是很好的.

但是目前你可以在命令行中调用 Matlab 运行自己的 Matlab 脚本, 一个测试脚本如下:

```
x = 1:0.1:2*pi;
y = sin(x);

figure('visible','off')
```

```
plot(x, y, 'r-');

exportgraphics(gcf, 'myfig.pdf')
```

然后在命令行中使用如下命令进行运行:

```
matlab -batch "run('matlab.m')"
```

运行完后，你便可以在当前目录下看到一个 pdf 文件，名为 `myfig.pdf`; 在运行方式这一点上,Matlab 和 Wolframscriprt 的运行命令:`wolframscript -script mma.wls` 是有一点区别的.

1.9.2 现状

似乎在 TEX 中无法正常的调用 Matlab 进行计算与绘图，Matlab 日志显示错误. 但是目前此功能应该不会在短时间内开发出来. 如果有比较多的需求，我可能会考虑.

1.10 数值计算

1.10.1 xfp

众说周知，TEX 自身的计算能力是比较羸弱的，所以涉及到一定的计算需求时，一般宏包的解决方法都是使用外部程序，让 TEX 只负责排版就行了. 但是在 LATEX3 项目发展了这么久之后，也做出了一些令人惊喜的结果. 这里我们主要介绍 LATEX3 的 xfp 宏包，用于浮点数运算. 这里说明部分 xfp也许可以解决的痛点:

- 在 TikZ 绘图中，常常是需要坐标运算的，尽管 TikZ 提供了一个 calc 库, 但是这个库的使用语法总觉得不是那么的自然. 于是这个时候你就可以使用 xfp 宏包.
- 在你自定义一些需要用到数值计算的宏命令时，使用 xfp 宏包是一个比较好的选择.

xfp 宏包的详细使用教程请参见其官方文档, 这里不再对此宏包进行过多的描述. zTikZ 或者是 zLATEX 并不会自动加载 xfp 宏包，如果你有这方面的需要，请用户自行加载.

1.10.2 python

上面介绍了 Python 的绘图功能，所以我们在这里 zTikZ 中的浮点数计算功能. 一方面，用户可以使用 python 模块的 sympy 库进行浮点数计算, 亦或者是使用其进行符号计算. zTikZ 中的浮点数运算功能主要基于 Python 中的 numpy, sympy 等库, zTikZ 在调用此命令时默认载入 Python 的 numPy, scipy 库. 此外用户在使用 numpy 中的函数时不用再加以前缀, 比如求解 sin(2.345) 时，直接使用`\py{sin(2.345)}` 即可, 不用写成`\py{np.sin(2.345)}`. 对于 `scipy` 库中的函数, 使用方法同理.

py command

zTikZ 提供了命令\py 用于浮点数运算, 这部分的结果并不会被缓存，也就是说每次编译此文档时，Python 都会重新计算此部分的结果. \py 的参数说明如下:

```
\py[<return type>]{<expression>}
```

上述的第一个默认参数值为:raw, 可选值有 str, 二者的区别可以简单的认为，返回 token 的类别码不同. 比如当外部文件中的内容为:

```
\[ a^2 + b^2 = c^2 \]
```

默认情况才下，\py 返回此命令的结果为:

$$a^2 + b^2 = c^2$$

但是如果你指定返回的类型为 str 时, 那么在文档中的显示结果就会变为:\[a^2 + b^2 = c^2 \]. 而不是默认情况下的排版公式.

值得说明的是，\py 命令和 xfp 宏包提供的\inteval ，\fpeval 是类似的；也就是你可以把 \py 命令嵌套到你自己定义的一个宏命令中. 同样是使用 #1 来表示接收到的参数. 比如你可以创建下面这几个命令:

```
\newcommand{\pypow}[1]{\py{#1}}
\newcommand{\pyreverse}[1]{\py{'#1'[::-1]}}
\newcommand{\pyuppercase}[1]{\py{'#1'.upper()}}
```

分别用于数值计算 (乘方计算), 字符串反转输出, 字符串大写输出. 使用效果如下:

- Power Calculation: $2^{10} = 1024$
- Reverse a string using Python: XeTaL-olleH
- Uppercase a string: HELLO-LATEX
- Modulus: $102 = 6 \mod 8$
- Return string Options: $$1024$$

如果你想要使用 Python 中的求模运算需要输入% 时, 在\py 命令中你应该写为:

```
\py{102\%8}
```

或者是如果你需要在\py 命令中传入字符:$, 请像下面这样书写:

```
\py{'\$\$'+str(2**10)+'\$\$'}
```

毕竟在命令行中，如下的命令才可以正常的工作:

```
sed -i "6s|Float_res = .*|Float_res = '\$\$'+str(2**10)+'\$\$'|"
./ztikz_output/scripts/python_script.py
```

注记 1.10.1 目前由于 Windows 上的 sed 命令 (又或者是平台差异) 和 Linux 下的差异, 所以可能导致在 Windows 上使用时，`\py` 中的单引号`'` 不能正确的输入到目标文件中, 从而导致字符串的声明失败. 请一定注意!

pycode environment

zTikZ 同时也提供了一个用于自由书写 Python 代码的环境 `pycode`, 可以用于生成复杂且规律的表格代码等排版元素. 比如下面的示例:

```
\begin{pycode}{pycode_1.py}
import numpy as np

# write file
with open ('./ztikz_output/python_data/pycode_1.py.out', 'w') as file:
    file.write("\\begin{tabular}{p{3cm}ccc}\n")
    file.write("\\hline\n")
    file.write("number/function & $\\sin$ & $\\cos$ & $\\tan$\\\\\n")
    file.write("\\hline\n")
    for i in range(1, 21):
        file.write(
            f"${i}$ & ${np.around(np.sin(i), decimals=4)}$ &  ${np.around(np.cos(i),
            decimals=4)}$ & ${np.around(np.tan(i), decimals=4)}$\\\\\n"
        )

    file.write("\\hline\n")
    file.write("\\end{tabular}\n")
\end{pycode}
```

那么在运行此命令后，在 zTikZ 的缓存文件夹中会生成一个名为 `pycode_1.py.out` 的文件，其内容为:

```
\begin{tabular}{p{3cm}ccc}
\hline
number/function & $\sin$ & $\cos$ & $\tan$\\
\hline
$1$ & $0.8415$ &  $0.5403$ & $1.5574$\\
$2$ & $0.9093$ &  $-0.4161$ & $-2.185$\\
$3$ & $0.1411$ &  $-0.99$ & $-0.1425$\\
$4$ & $-0.7568$ &  $-0.6536$ & $1.1578$\\
$5$ & $-0.9589$ &  $0.2837$ & $-3.3805$\\
$6$ & $-0.2794$ &  $0.9602$ & $-0.291$\\
$7$ & $0.657$ &  $0.7539$ & $0.8714$\\
$8$ & $0.9894$ &  $-0.1455$ & $-6.7997$\\
$9$ & $0.4121$ &  $-0.9111$ & $-0.4523$\\
$10$ & $-0.544$ &  $-0.8391$ & $0.6484$\\
$11$ & $-1.0$ &  $0.0044$ & $-225.9508$\\
$12$ & $-0.5366$ &  $0.8439$ & $-0.6359$\\
$13$ & $0.4202$ &  $0.9074$ & $0.463$\\
$14$ & $0.9906$ &  $0.1367$ & $7.2446$\\
$15$ & $0.6503$ &  $-0.7597$ & $-0.856$\\
\hline
\end{tabular}
```

然后把用 Python 生成的这段表格代码排版出来，具体效果如下:

number/function	sin	cos	tan
1	0.8415	0.5403	1.5574
2	0.9093	−0.4161	−2.185
3	0.1411	−0.99	−0.1425
4	−0.7568	−0.6536	1.1578
5	−0.9589	0.2837	−3.3805
6	−0.2794	0.9602	−0.291
7	0.657	0.7539	0.8714
8	0.9894	−0.1455	−6.7997
9	0.4121	−0.9111	−0.4523
10	−0.544	−0.8391	0.6484
11	−1.0	0.0044	−225.9508
12	−0.5366	0.8439	−0.6359
13	0.4202	0.9074	0.463
14	0.9906	0.1367	7.2446
15	0.6503	−0.7597	−0.856

表 1.2: Using Python to generate Table

注记 1.10.2 本环境 (`pycode`) 目前还不够成熟, 请谨慎使用, 也欢迎各位提出宝贵的改进意见. 当然, 本环境支持 cache 模块.

注记 1.10.3 推荐用户使用最新的由 LaTeX3 编写的宏包:csvsimple-l3, 或者是tabularray用于 LaTeX 中的表格排版或者表格数据的读取等操作.

1.10.3 mathematica

使用 Mathematica 进行数值计算这一部分和后面的`\wolfram` 指令是有一部分重合的, 详细的使用教程请参见:节 1.11符号计算, 所以这一部分我们就在后面介绍.

1.11 符号计算

符号计算是区别于数值计算的，上述的数值计算章节应该也有介绍; 但在介绍 zTikZ 中的符号计算模块之前先给出一个符号计算的定义，以下定义摘自 wiki:

> 数学和计算机科学中，计算机代数或符号计算或代数计算，是研究、开发用于操作表达式等数学对象的算法与软件的科学领域. 这通常被视为是运算科学的一个子领域，但运算科学一般基于近似浮点数的数值计算，而符号计算则使用含变量的表达式进行精确计算，其中变量没有赋值. 执行符号计算的软件系统称为计算机代数系统 (computer algebra system, CAS), “系统”暗示了软件的复杂性，其中至少包括一种在计算机中表示数学数据的方法、一种编程语言（通常异于用于实现的语言）、一种专门的内存管理器、一套供输入输出表达式的用户界面、一大套用于通常运算的子程序，如表达式简化、能实现链式法则、多项式因式分解、不定积分等等的求导算法.

当前流行的计算机代数系统主要有:

- mathHandbook.com
- Sagemath
- Mathematica
- Maple
- MAGMA
- Maxima
- GAP
- PARI/GP
- Meditor
- MuPAD
- Mathomatic
- Xcas/Giac
- Yacas
- Mate

zTikZ 主要提供一个和 Wolfram/Mathematica(假如你已经购买了该软件) 或者 Pyhton 的 sympy 进行交互的模块，以此提供对应的符号计算功能, 后续可能会引入 octave 的接口.

1.11.1 sympy

Python 的 sympy 是一个**免费，开源，轻量**的符号计算模块，其官网上有着详细的教程. 所以这里便不再赘述其语法，重点介绍 zTikZ 中提供的几个接口 (命令)，用于和 sympy 交

互. zTikZ 中针对 sympy 提供了命令:`\sympy` ，其参数格式为:

```
\sympy{<expression>}
```

和之前使用 Python 用于数值计算不同的是:zTikZ 针对此命令提供了 cache 机制，此命令对应的结果会被保存在文件: `./ztikz_output/python_data/sympy_<index>.out`. 此文件名中的 `<index>` 表示的是对应的符号运算序号 (第几次符号运算的结果).

`\sympy` 命令的运算结果被保存在文件中之后，通过`\input` 命令把对应的运算结果导入到 TEX 的输出流 (文档) 中，由于默认的情况下此结果包含数学公式中的上下标:`^`，`_`，... 等, 所以在把其导入到 LATEX 源码中时需要放入数学环境中.

zTikZ 模块的`\sympy` 命令在进行符号运算时, 默认的符号变量有:`x`，`y`，`z`，`u`，`v`，`t`, 这些预定义变量无需用户再次声明，用户可直接使用; 下面给出使用`\sympy` 命令进行符号计算的部分示例:

```
% 定积分
\sympy{integrate(sin(x)/x, (x, -oo, oo))}
% 不定积分
\sympy{integrate( x**8 + cos(7*x) + 6*t, x )}
% 矩阵特征值
\sympy{Matrix([[1, 2], [2, 2]]).eigenvals()}
% 极限计算
\sympy{limit(sin(x)/x, x, 0)}
```

计算定积分的例子:

$$\int_{-\infty}^{+\infty} \frac{\sin(x)}{x}\, dx = \pi$$

或者是计算不定积分的例子:

$$\int x^8 + \cos(7x) + 6t \, dx = 6tx + \frac{x^9}{9} + \frac{\sin(7x)}{7}$$

或者是一个计算特征值的例子:

$$\mathrm{eig}(\begin{bmatrix} 1 & 2 \\ 2 & 2 \end{bmatrix}) = \left\{ \frac{3}{2} - \frac{\sqrt{17}}{2} : 1, \ \frac{3}{2} + \frac{\sqrt{17}}{2} : 1 \right\}$$

计算极限的例子:

$$\lim_{x \to 0} \frac{\sin x}{x} = 1$$

目前的`\sympy` 命令只支持单行命令的模式，如果你需要使用多行 (条) 命令来达到计算目的，请考虑把它们变为一行命令 (一条指令).

1.11.2 mathematica

zTikZ 模块提供和 Mathematica 相关的符号计算，数值运算和知识查询接口; 以下的所有命令均具有缓存机制.

- `\wolfram[<option>]{<expression>}`: 使用 Mathematica 计算此表达式 `<expression>`, 默认返回 TEX 格式的代码，可以把 `<option>` 设为 `text`, 让其返回一个文本对象. 可以在这个命令中执行任何的 wolfram 指令，但是需要注意的一点是，所有和 wolfram 相关的命令是不会自动进入数学模式的，需要手动添加数学模式的标记.
- `\wolframSolve[<cmd-style>]{<expression>}[<varible>][<domain>]`: 其中第一个可选参数默认值为: `part`, 意味着你的命令需要分拆为 3 个部分: 表达式 – (求解) 变量名 – 求解范围，对应上面的参数，分别填入. 如果指定第一个参数为 `full`, 那么此时只需要给出对应的 `<expresion>`, 不用再次指定后续参数.(毕竟在第二个 (强制性-Mandatory) 参数中就已经包含了这些信息, 参见后面的具体使用样例).
- `\wolframDSolve[<cmd-style>]{<equation>}[<independent-varible>][<dependent-variablei>]`: 此命令用于求解微分方程，其中的第一个可选参数和上面的`\wolframSolve` 的意义一致, 不再赘述. 第二个参数表示要求解的微分方程，第三个参数表示求解的独立变量 (函数)，最后一个参数表示此微分方程求解函数的自变量.

wolfram

此命令会把求解结果保存到一个临时变量`\wolframResult` 中，所以可以不用把此命令置于公式环境中, 直接在公式环境中使用命令`\wolframResult` 引用对应的结果即可, 引用方式有两种:

- `\wolframResult[raw][<separator>]`: 直接引用全部结果 (默认不同的结果使用 “,” 进行分割)，不同的结果之间使用 `<separator>` 进行分割. 第一个可选参数默认为 `raw`, 也就是你可以直接使用 `\wolframResult` 引用上一步的计算结果.
- `\wolframResult[<integer-expresion>]`: 引用单个结果，可以是任何合法的整数运算表达式.

在这里给出`\wolfram` 命令的部分使用样例:

```
\wolfram{Series[Exp[x], {x, 0, 5}]}
\[\wolframResult\]

\wolfram{LaplaceTransform[t^4 Sin[t],t,s]}
\[\wolframResult\]

\wolfram[text]{WolframAlpha["Shanghai Population", "ShortAnswer"]}
\[\wolframResult\]
```

函数 $y = \mathrm{e}^x$ 的 5 阶 Taylor 展开式为:

$$1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120} + O\left(x^6\right)$$

函数 $x = t^4 \sin(t)$ 的 Laplace 变换为:

$$\mathcal{L}[t^4 \sin(t)] = \frac{24\left(5s^4 - 10s^2 + 1\right)}{\left(s^2+1\right)^5}$$

在\wolfram 指令中执行 Mathematica 中的 WolframAlpha 命令进行查询，比如这里查询上海的人口数量，结果为:about 24.9 million people

这里补充一个使用\wolfram 就行数值运算的例子，因为 Mathematica 中有着诸多和数值运算相关的函数，这里仅以内置的函数 N[<expression>] 为例:

比如我们求解 π 的截取前 30 小数的近似值为:

$$\pi \approx 3.14159265358979323846264338328$$

> 在使用\wolfram 命令进行浮点数运算时，只要表达式中含有小数，那么 Mathematica 就会默认进行浮点数运算，而不会计算表达式的精确值.

wolframSolve

\wolframSolve 命令可以用于多项式方程根的求解以及方程组的求解，并且可以给定求解的范围. 和前面的\wolfram 命令类似, 下面给出几个比较简单的求解示例:

```
\wolframSolve{x^4 - x^2 - 5 == 0}[x]
\begin{align}
    \left\{\begin{aligned}
        & \wolframResult[raw][\\ &]
    \end{aligned}\right.
\end{align}

\wolframSolve{a x + y == 7 && b x - y == 1}[x, y]
\[\wolframResult[raw][,\;]\]

\wolframSolve{x^2 + 2 y^3 == 3681 && x > 0 && y > 0}[x, y][Integers]
\[\wolframResult[raw][\;||\;]\]

\wolframSolve[full]{x^2 + y^2 == 5^2 && y > x > 0, {x, y}, Integers}
\[\wolframResult[raw][\leftrightarrow]\]
```

方程 $x^4 - x^2 - 5 = 0$ 的所有根为:

$$\begin{cases} x = -i\sqrt{\frac{1}{2}\left(\sqrt{21}-1\right)} \\ x = i\sqrt{\frac{1}{2}\left(\sqrt{21}-1\right)} \\ x = -\sqrt{\frac{1}{2}\left(1+\sqrt{21}\right)} \\ x = \sqrt{\frac{1}{2}\left(1+\sqrt{21}\right)} \end{cases} \tag{1.11.1}$$

方程组 $\begin{cases} ax + y = 7 \\ bx - y = 1 \end{cases}$ 的解为:

$$x = \frac{8}{a+b},\ y = -\frac{a-7b}{a+b}$$

不定方程 $\begin{cases} x^2 + 2y^3 = 3681 \\ x > 0, y > 0 \end{cases}$ 的整数解为:

$$x = 15 \mid\mid y = 12 \mid\mid x = 41 \mid\mid y = 10 \mid\mid x = 57 \mid\mid y = 6$$

不定方程 $\begin{cases} x^2 + y^2 = 5^2 \\ x > y > 0 \end{cases}$ 的整数解为:

$$x = 3 \leftrightarrow y = 4$$

目前解的筛选功能仅支持类似数组的访问形式，引用形式比较有限．背后其实就是根据不同解之间分隔符 ‘,’，把返回结果中的 TokenList 分割为一个 sequence．根据划分的结果生成一个 sequence 列表，然后采用一个整数进行索引．

wolframDSolve

命令\wolframDSolve 和命令\wolframSolve 完全相同，只是这个命令是用于求解微分方程的．下面是几个示例:

```
\wolframDSolve{{y'[x] + y[x] == a*Sin[x], y[0] == 0}}[y[x]][x]
\wolframDSolve[full]{{y'[x] == Exp[z[x]] + 1, z'[x] == y[x] - x}, {y[x], z[x]}, x}
```

微分方程 $y' + y = a\sin(x), y(0) = 0$ 的解为:

$$y(x) = -\frac{1}{2}ae^{-x}\left(-e^x\sin(x) + e^x\cos(x) - 1\right)$$

非线性系统微分方程组 $y'(x) + y(x) = \mathrm{Exp}(z(x)) + 1, z'(x) = y(x) - x$ 的解为:

$$\begin{cases} z(x) = \log\left(c_1\tan^2\left(\frac{1}{2}\left(\sqrt{2}\sqrt{c_1}x + 2\sqrt{2}\sqrt{c_1}c_2\right)\right) + c_1\right) \\ y(x) = x + \sqrt{2}\sqrt{c_1}\tan\left(\frac{1}{2}\left(\sqrt{2}\sqrt{c_1}x + 2\sqrt{2}\sqrt{c_1}c_2\right)\right) \end{cases} \tag{1.11.2}$$

1.12 Plot Gallery

1.12.1 tikz

下面我们给出 zTikZ 这部分命令的一些综合绘图案例:

如果你修改了绘图代码，但是发现得到的 pdf 中的图像并没有改变，那么极有可能是因为你指定的精度过高，超出了 TEX 的内存使用限制.(而且由于采用了 external 库用于缓存，有可能你在编译时并不会抛出这个错误) 其实比较耗费内存的点主要有 3 个:

- 指定的精度过高，一般情况下在区间长度 < 5 时指定精度为 100 就已经足够了
- 使用了多个`\ContourPlot` 函数，在默认的精度 100 下，多个此函数也可能导致内存超出
- 最后一点耗时的点就是`\ShowIntersecion` 命令，可以先用 Geogebra 得到交点后再使用 `\ShowPoint` 命令进行点的绘制.
- 更严重的如果出现了编译错误，请考虑去掉`\ShadePlot` 命令，或在`\usepackage [external=false]{ztikz}` 的情况下使用此命令.

Example 1

```
% show angle
\tikzset{
    right angle quadrant/.code={
        \pgfmathsetmacro\quadranta{{1,1,-1,-1}[#1-1]}
        \pgfmathsetmacro\quadrantb{{1,-1,-1,1}[#1-1]}},
    right angle quadrant=1,
    right angle length/.code={\def\rightanglelength{#1}},
    right angle length=2ex,
    right angle symbol/.style n args={3}{
        insert path={
            let \p0 = ($(#1)!(#3)!(#2)$) in
                let \p1 = ($(\p0)!\quadranta*\rightanglelength!(#3)$),
                \p2 = ($(\p0)!\quadrantb*\rightanglelength!(#2)$) in
                let \p3 = ($(\p1)+(\p2)-(\p0)$) in
            (\p1) -- (\p3) -- (\p2)
        }
    }
}
% main code
\begin{tikzpicture}[scale=.75, >=Latex]
    \ShowGrid[step=1, color=gray, opacity=.5]{(-8, -8); (8, 8)}
```

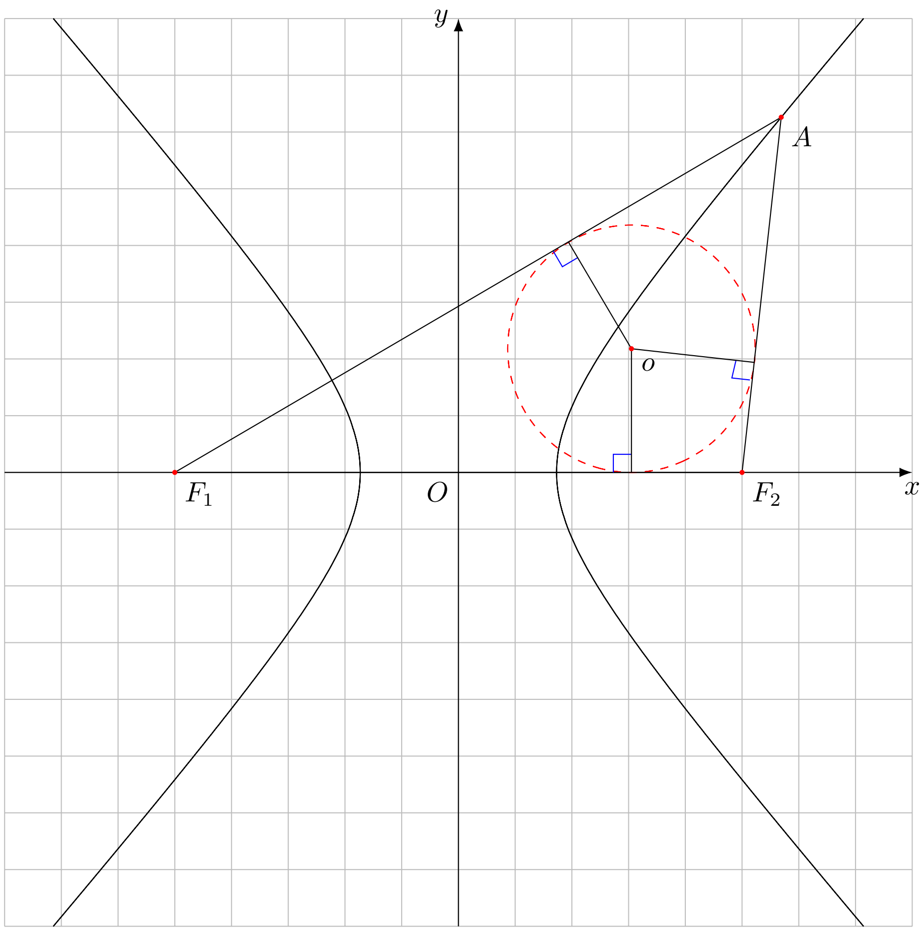


图 1.19: Parabolic Example

```
\ShowXYAxis{8}{8}
% curve
\ContourPlot[-8:8; -8:8]{x**2/3-y**2/4-1}
\ContourPlot[-8:8; -8:8][dashed, red]{(x-3.05)**2 + (y-2.18)**2-4.76}
% points and lines
\coordinate (A) at (5.69, 6.26);
\coordinate (B) at (-5, 0);
\coordinate (C) at (5, 0);
\coordinate (D) at (3.05, 2.18);
\draw (A) -- (B) -- (C) -- cycle;
% angle and 3-lines
\draw [blue,right angle symbol={A}{1.94,4.06}{D}]; % F
\draw [blue,right angle symbol={A}{5.21,1.94}{D}]; % E
\draw [blue,right angle symbol={B}{3.05,0}{D}];    % G
\draw (D) -- (1.94, 4.06);
\draw (D) -- (5.21, 1.94);
\draw (D) -- (3.05, 0);
```

```
    \ShowPoint[type=*, color=red]{(A); (B); (C); (D)}[$A$; $F_1$; $F_2$; $o$][below
    right]
\end{tikzpicture}
```

Example 2

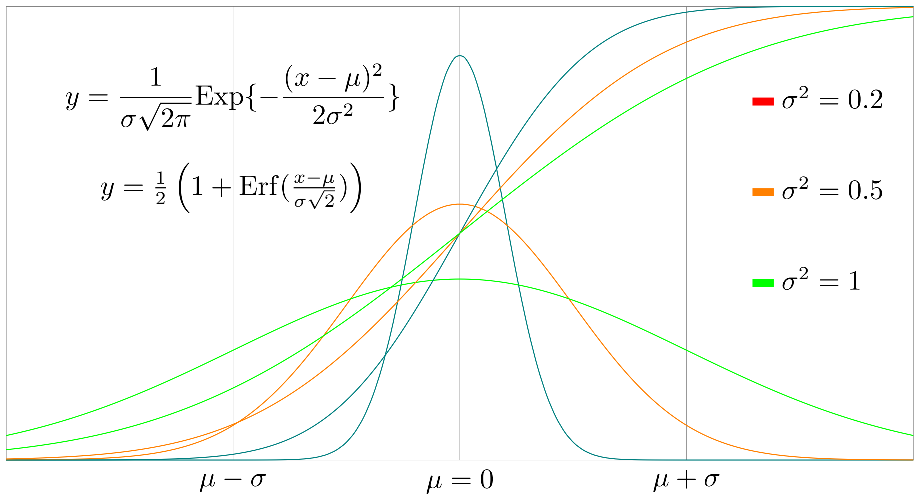


图 1.20: Normal-distribution Example

```
\begin{tikzpicture}[yscale=6, xscale=3]
    \ShowGrid{(-2,0); (2,1)}
    % pdf
    \Plot[-2:2][teal]{1/(sqrt(0.2)*sqrt(2*pi))*exp(-(x-0)**2/(2*0.2**2))}
    \Plot[-2:2][orange]{1/(sqrt(0.5)*sqrt(2*pi))*exp(-(x-0)**2/(2*0.5**2))}
    \Plot[-2:2][green]{1/(sqrt(1)*sqrt(2*pi))*exp(-(x-0)**2/(2*1**2))}
    % cdf
    \Plot[-2:2][teal]{0.5*(1+erf((x-0)/(sqrt(0.2)*sqrt(2))))}
    \Plot[-2:2][orange]{0.5*(1+erf((x-0)/(sqrt(0.5)*sqrt(2))))}
    \Plot[-2:2][green]{0.5*(1+erf((x-0)/(sqrt(1)*sqrt(2))))}
    % annotate
    \ShowPoint[radius=0pt]{(-1, 0); (0, 0); (1, 0)}[$\mu-\sigma$; $\mu=0$;
    $\mu+\sigma$][below]
    \ShowPoint[radius=0pt]{(1, 0.8); (1, 0.6); (1, 0.4)}[
        \textcolor{red}{\rule[1pt]{8pt}{3pt}}\;$\sigma^2=0.2$;
        \textcolor{orange}{\rule[1pt]{8pt}{3pt}}\;$\sigma^2=0.5$;
        \textcolor{green}{\rule[1pt]{8pt}{3pt}}\;$\sigma^2=1$;
    ][right=2em]
    \ShowPoint[radius=0pt]{(-1, 0.8)}[$\displaystyle y =
    \frac{1}{\sigma\sqrt{2\pi}}\mathrm{Exp}\{-\frac{(x-\mu)^2}{2\sigma^2}\}$]
    \ShowPoint[radius=0pt]{(-1,
    0.6)}[$y=\frac12\left(1+\mathrm{Erf}(\frac{x-\mu}{\sigma\sqrt{2}})\right)$]
\end{tikzpicture}
```

Example 3

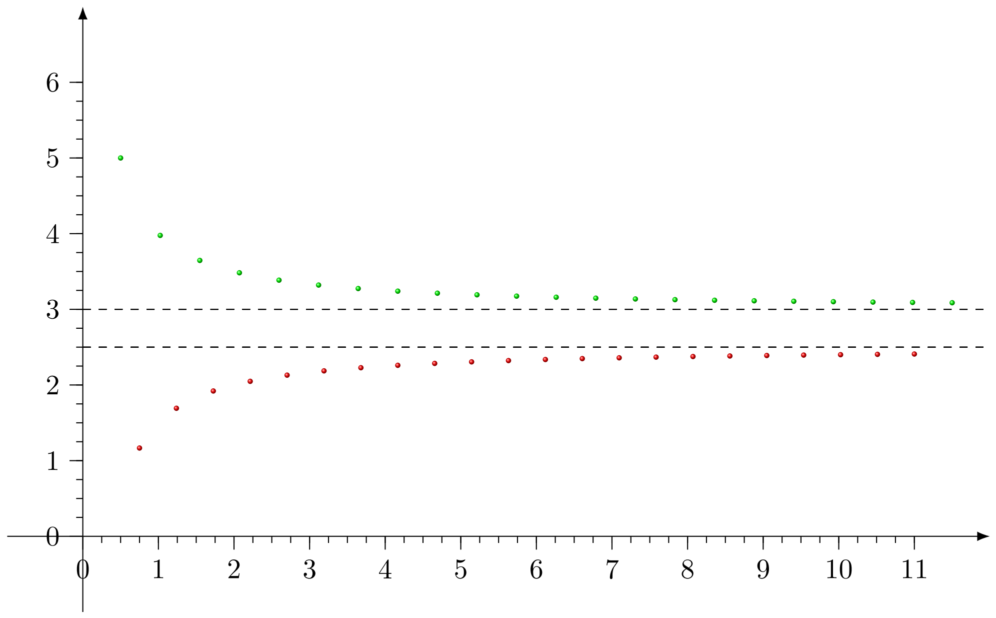


图 1.21: Sup and Inf Example

```
\begin{tikzpicture}[>=Latex]
    \xAxis[-1][12]
    \yAxis[-1][7]

    \PlotPrecise{plot}{22}
    \Plot[0.75:11][red, thick, opacity=0][type=ball, color=red]{2.5-1/x}

    \PlotPrecise{plot}{22}
    \Plot[0.5:11.5][red, thick, opacity=0][type=ball, color=green]{3+1/x}

    \PlotPrecise{contour}[long]{40}
    \ContourPlot[0:12][dashed]{y-2.5}
    \ContourPlot[0:12][dashed]{y-3}
\end{tikzpicture}
```

Example 4

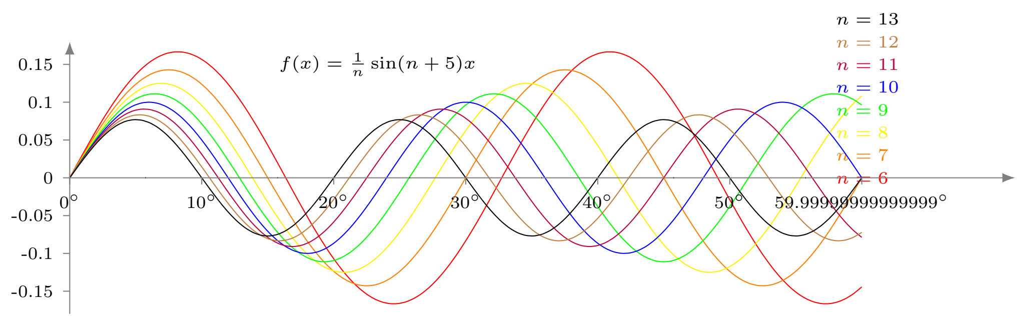


图 1.22: Loop Plot Example

```
\ExplSyntaxOn
\clist_new:N \l__color_clist
\clist_set:Nn \l__color_clist {red, orange, yellow, green, blue, purple, brown, black}
\newcommand{\colorItem}[1]{\clist_item:Nn \l__color_clist {#1}}
\ExplSyntaxOff
\begin{tikzpicture}[scale=10, >=Latex, font=\scriptsize]
    % plot and annotate
    \node at (.55, 0.15) [left] {$f(x)=\frac{1}{n}\sin(n+5)x$};
    \foreach \i in {6, 7, 8, 9, 10, 11, 12, 13}{
        \Plot[0:pi/3][\colorItem{\fpeval{\i-5}}]{\fpeval{1/\i}*sin(\fpeval{\i+5}*x)}
        \node[color=\colorItem{\fpeval{\i-5}}] at (1, \fpeval{(\i-6)*0.03}) [right]
        {$n=\i$};
    }
    % axis draw
    \ShowAxis [
        tickStyle=above,    axisColor=gray,
        tickStart=-0.15,    tickEnd=0.18,
        mainStep=0.05,
        mainTickColor=gray, mainTickLabelPosition=left,
        mainTickLength=.5pt,axisRotate=90,
    ]{(-0.18, 0); (0.18, 0)}
    \ShowAxis [
        tickStyle=below,    axisColor=gray,
        tickStart=0,        tickEnd=1.22,
        mainStep=\fpeval{3.1415926/18},
        mainTickColor=gray, subTickLength=0pt,
        mainTickLength=.5pt,
        mainTickLabel={\fpeval{\CurrentFp/(3.1415926/18)*10}$^\circ$}
    ]{(0, 0); (1.25, -0)}
\end{tikzpicture}
```

Example 5

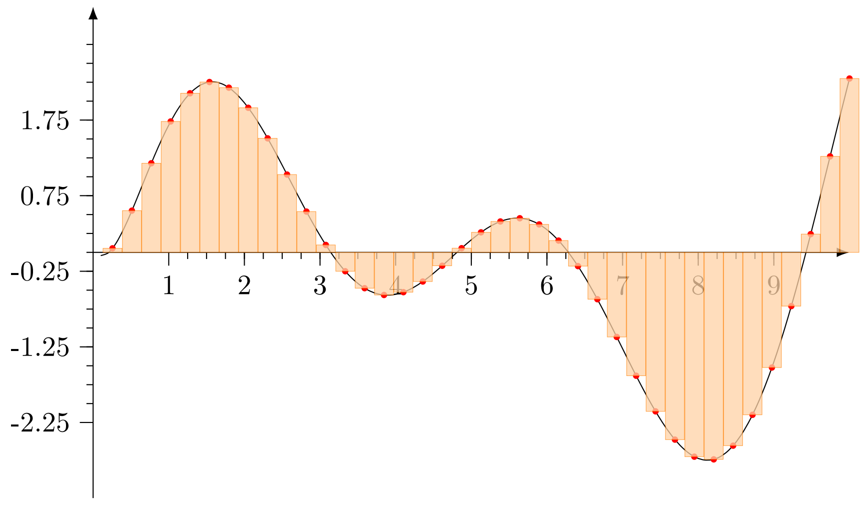


图 1.23: Darboux Example

```
\begin{tikzpicture}[>=Latex]
    % draw axis
    \xAxis[0][10]
    \yAxis[-3.25][3.25]
    % plot ucntion and  generate discrete points
    \Plot[0:10]{2*sqrt(x)*cos(log(x))*sin(x)}
    \PlotPrecise{plot}{40}
    \Plot[0:10][opacity=0][type=*, color=red]{2*sqrt(x)*cos(log(x))*sin(x)}
    % bar plot
    \BarPlot[x][fill=orange!35!white, bar width=\fpeval{10/40}cm, opacity=.75, very
    thin, draw=orange]{\gnudata{2}}
\end{tikzpicture}
```

Example 6

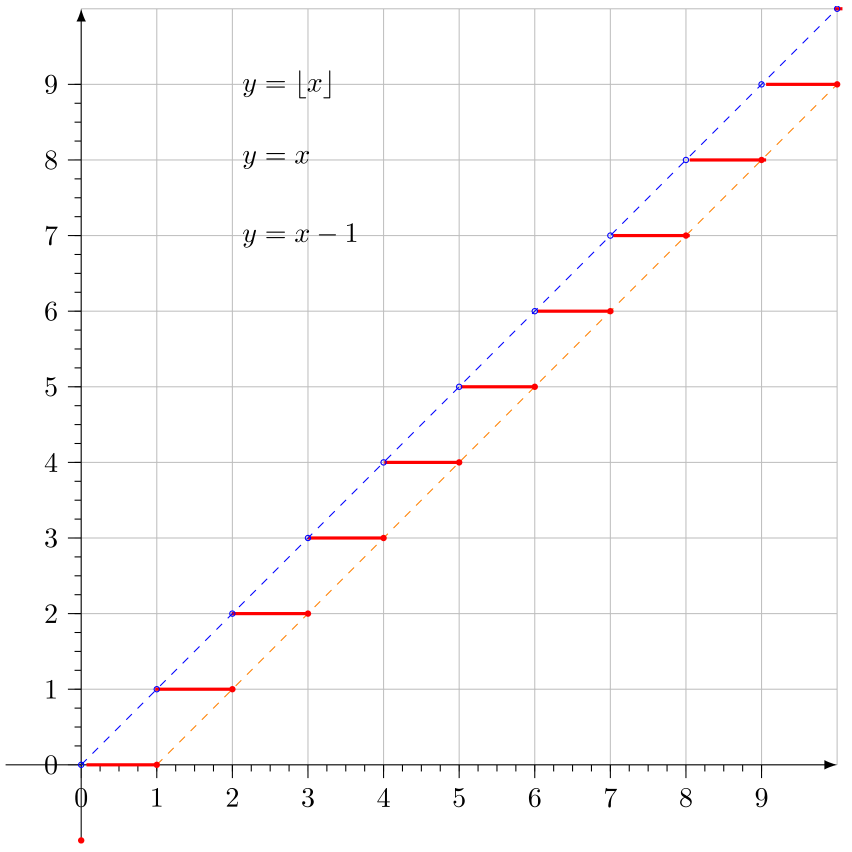


图 1.24: Stairs Function Example

```
\begin{tikzpicture}[>=Latex]
    % grid and axis
    \ShowGrid[step=1, color=gray, opacity=.5]{(0, 0); (10, 10)}
    \xAxis[-1][10]      \yAxis[-1][10]
    % plot and points
    \Plot[0:10][red, jump mark right, very thick, xshift=2pt][type=*,
    opacity=0]{floor(x)}
    \Plot[0:10][dashed, blue]{x}
    \Plot[1:10][dashed, orange]{x-1}
    \PlotPrecise{plot}{11}
    \Plot[0:10][opacity=0, jump mark right][type=o, color=blue]{x}
    \PlotPrecise{plot}{11}
    \Plot[0:10][opacity=0, jump mark right][type=*, color=red]{x-1}
    \ShowPoint[opacity=0]{(2, 9); (2, 8); (2, 7)}[$y=\lfloor x\rfloor$; $y=x$;
    $y=x-1$][right]
\end{tikzpicture}
```

Example 7

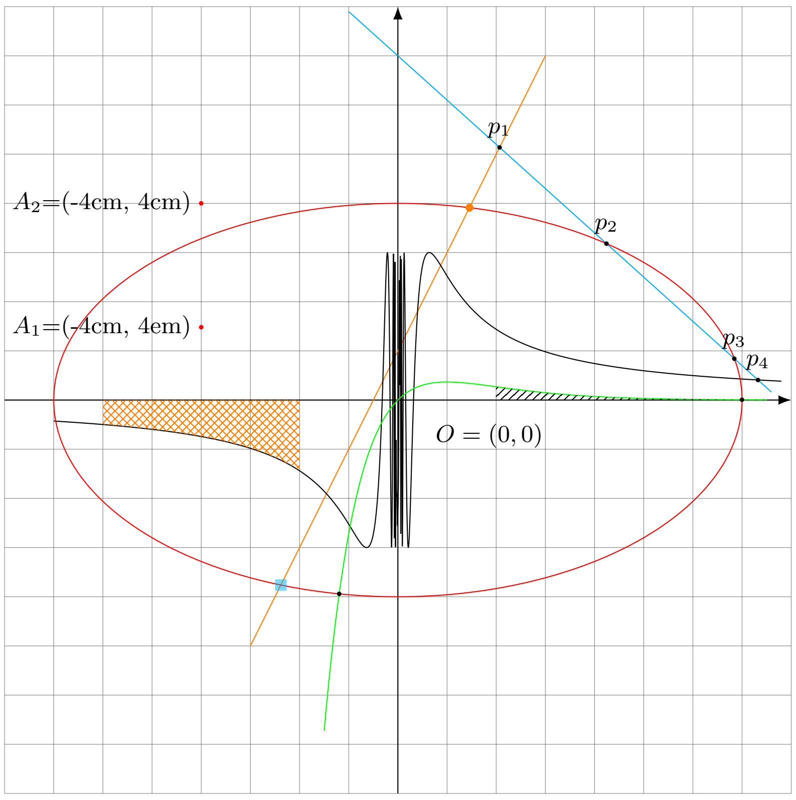


图 1.25: Functions and Points Example

```
\begin{tikzpicture}[font=\small, >=Latex, scale=.65]
    %% ==> ztikz draw grid(coordinates)
    \ShowGrid{(-8, -8); (8, 8)}
    \ShowPoint[opacity=0]{(0, 0)}[$O=(0, 0)$][below right=.5em and 1em]
    \ShowAxis {(0, -8); (0, 8)}
    \ShowAxis {(-8, 0); (8, 0)}

    % draw function
    \Plot[-3:3][orange] {2*x+1}
    \Plot[-1:7.6][cyan] {-.9*x+7}

    %% ==> ztikz Plot functions
    % simple 2d-plot
    \PlotPrecise{plot}{1500}
    \Plot[-7:7.8]{3*sin(1/x)}
    \Plot[-1.5:7.5][green, name path=exp] {x*exp(-x)}
    % param plot
    \ParamPlot[0:2*pi][red, name path=ellipse]{7*sin(t), 4*cos(t)}
    %%% ==> Fill region
    \begin{scope}
        \clip (2, 0) rectangle (8, 1);
        \fill[pattern=north east lines] plot file{\gnudata{4}};
    \end{scope}
    \begin{scope}
        \clip (-6, 0) rectangle (-2, -2);
        \fill[pattern=crosshatch, pattern color=orange] plot file{\gnudata{3}} -- (-2,
        0) -- (-6, 0);
    \end{scope}

    % default tikz unit
    \ShowPoint[color=red]{(-4cm, 4em); (-4cm, 4cm)}[$A_1$=(-4cm, 4em); $A_2$=(-4cm,
    4cm)][left]

    % plot shape and other properties
    \ShowPoint[radius=3pt, color=blue, opacity=.5, type=square*, color=cyan] {(-2.380,
    -3.761)}
    \ShowPoint[color=orange, opacity=1, radius=2pt] {(1.456, 3.912)}

    % ==> find intersection
    \ShowIntersection{exp; ellipse}{2}
    \ShowPoint {(2.068, 5.137); (4.242, 3.181); (6.843, 0.840); (7.324, 0.408)}
        [$p_1$; $p_2$; $p_3$; $p_4$; $p_5$; $p_6$; $p_7$][above]
\end{tikzpicture}
```

Example 8

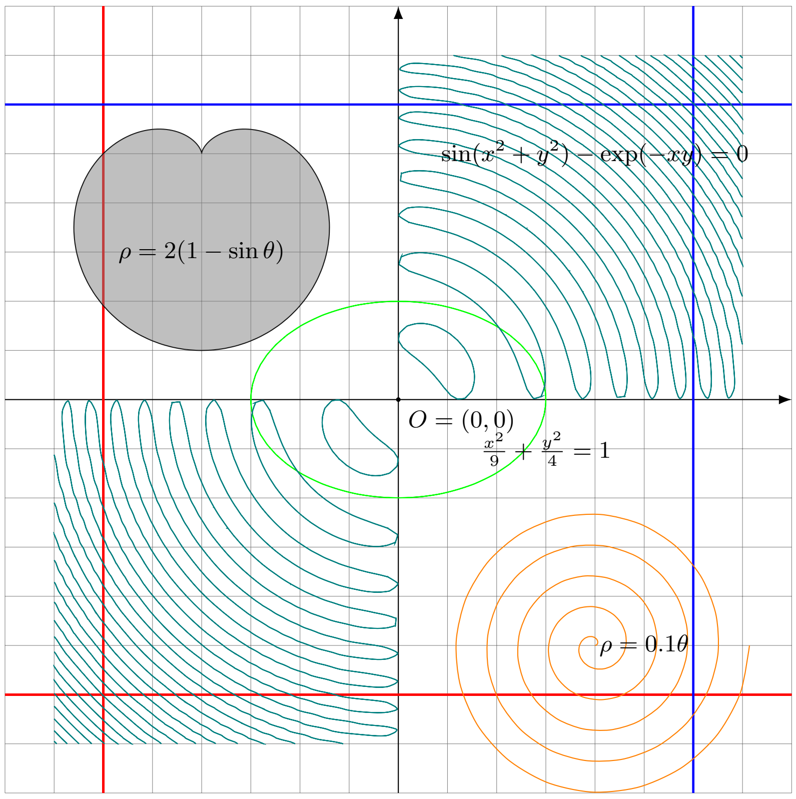


图 1.26: Polar and Contour Example

```
\begin{tikzpicture}[font=\small, >=Latex, scale=.65]
    %% ==> ztikz draw grid(coordinates)
    \ShowGrid{(-8, -8); (8, 8)}
    \ShowPoint{(0, 0)}[$O=(0, 0)$][below right]
    \ShowAxis {(0, -8); (0, 8)}
    \ShowAxis {(-8, 0); (8, 0)}

    % contour plot
    \PlotPrecise{contour}[long]{10}
    \ContourPlot[-8:8; -8:8][thick, red]{x+6}
    \ContourPlot[-8:8; -8:8][thick, red]{y+6}
    \ContourPlot[-8:8; -8:8][thick, blue]{y-6}
    \ContourPlot[-8:8; -8:8][thick, blue]{x-6}
    \PlotPrecise{contour}[long]{100}
    \ContourPlot[-4:4][green]{x**2/9+y**2/4-1}
    \ContourPlot[-7:7; -7:7][teal]{sin(x**2+y**2)-exp(-x*y)}

    % polar plot
    \begin{scope}[xshift=4cm, yshift=-5cm]
        \PolarPlot[0:10*pi][orange]{0.1*t}
    \end{scope}
    \begin{scope}[xshift=-4cm, yshift=5cm]
        \PolarPlot{2*(1-sin(t))}
        \fill[gray, opacity=.5] plot file {\gnudata{8}};
    \end{scope}

    % node annotate
    \ShowPoint[opacity=0]{(-4, 3); (4, 5); (5, -5); (3, -1)}
        [$\rho=2(1-\sin\theta)$; $\sin(x^2+y^2)-\exp(-xy)=0$; $\rho=0.1\theta$;
        $\frac{x^2}{9} + \frac{y^2}{4}=1$]
\end{tikzpicture}
```

Example 9

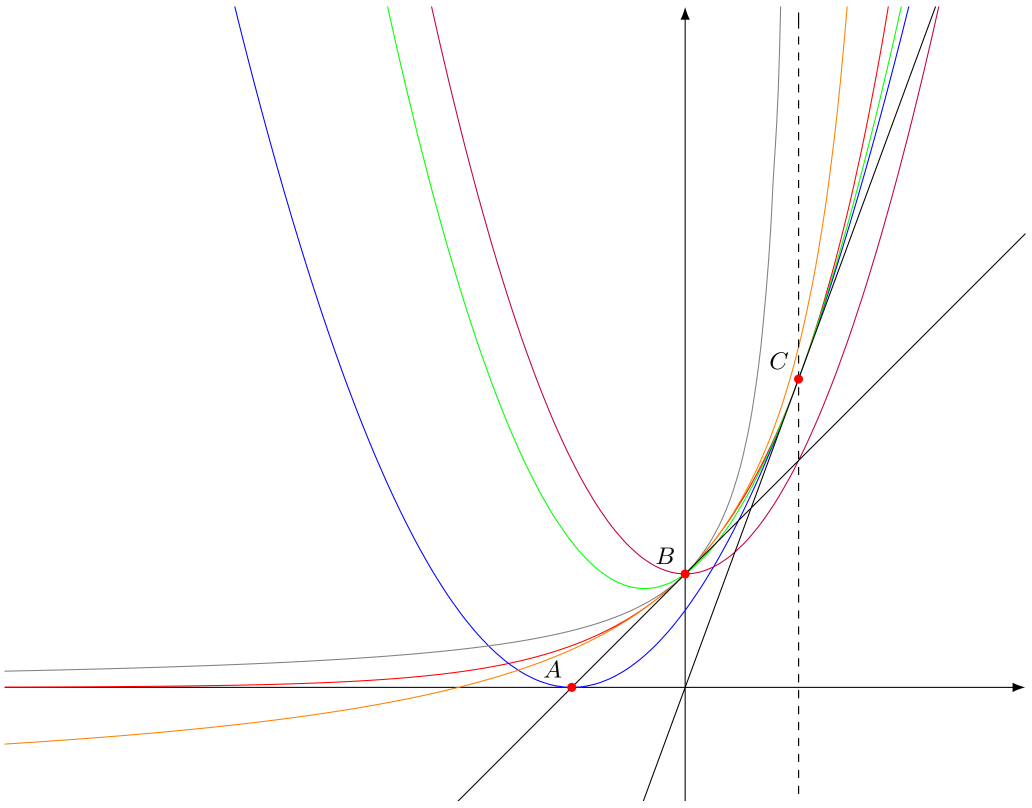


图 1.27: Exp Functions Example

```
\begin{tikzpicture}[>=Latex, font=\small, scale=1.5]
    % basic
    \clip (-6, -1) rectangle (3, 6);
    \ShowAxis{(-8, 0); (3, 0)}      \ShowAxis{(0, -1.5); (0, 6)}
    % functions
    \Plot[-8:5][red]                {exp(x)}
    \Plot[-8:5][blue]               {exp(1)/4*(x+1)**2}
    \Plot[-8:5][green]              {exp(1)*x + (x-1)**2}
    \Plot[-8:5][purple]             {x**2 + 1}
    \Plot[-8:0.95][gray]            {1/(1-x)}
    \Plot[-8:1.95][orange]          {(2+x)/(2-x)}
    \Plot[-8:5]                     {x+1}
    \Plot[-8:8]                     {exp(1)*x}
    \ContourPlot[0:2;-6:6][dashed]  {x-1}
    % points
    \ShowPoint[color=red, radius=1pt]{(-1, 0); (0, 1); (1, 2.71828)}[$A$; $B$;
    $C$][above left]
\end{tikzpicture}
```

Example 10

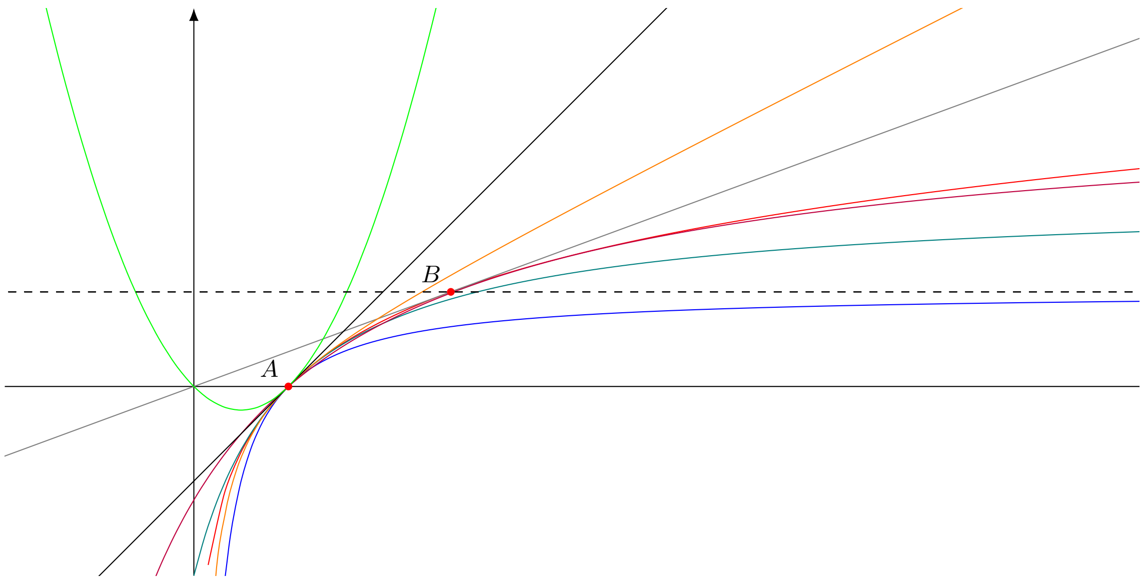


图 1.28: Log Functions Example

```
\begin{tikzpicture}[>=Latex, font=\small, scale=1.25]
    % basic
    \clip (-2, -2) rectangle (10, 4);
    \ShowAxis{(-2, 0); (12, 0)}     \ShowAxis{(0, -2); (0, 4)}
    % functions
    \Plot[-5:12][red]               {log(x)}
    \Plot[0:12][blue]               {(x-1)/x}
    \Plot[0:12][teal]               {2*(x-1)/(x+1)}
    \Plot[-1:12][purple]            {6*(x-1)/(2*x+5)}
    \Plot[-5:12][gray]              {x/exp(1)}
    \Plot[0.1:12][orange]           {0.5*(x-1/x)}
    \Plot[-5:12]                    {x-1}
    \Plot[-5:12][green]             {x**2-x}
    \ContourPlot[-5:12;-6:6][dashed]{y-1}
    % points
    \ShowPoint[color=red, radius=1pt]{(1, 0);(2.71828, 1)}[$A$; $B$][above left]
\end{tikzpicture}
```

Example 11

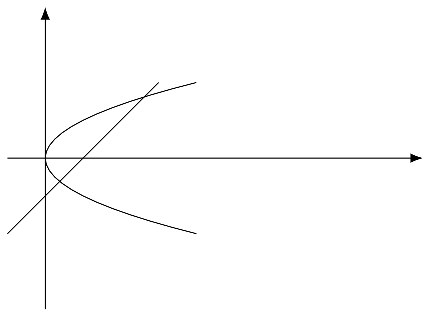


图 1.29: 11 Inverse Function Plot-1

```
\begin{tikzpicture}[scale=0.5, >=Latex]
    % axis
    \draw[->](-1,0)--(10,0);
    \draw[->](0,-4)--(0,4);
    % plot function
    \draw[domain=-2:2] plot({(\x)^2}, \x);
    \draw[domain=-2:2] plot(\x+1, \x);
\end{tikzpicture}
```

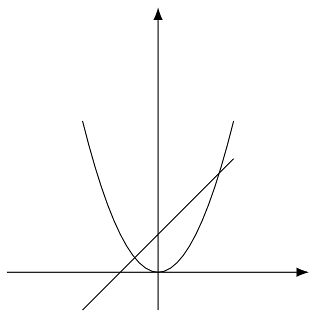


图 1.30: 11 Inverse Function Plot-2

```
\begin{tikzpicture}[scale=0.5, >=Latex]
    % axis
    \draw[->](0, -1) -- (0, 7);
    \draw[->](-4, 0)   -- (4, 0);
    % plot function
    \draw[domain=-2:2] plot(\x, {(\x)^2});
    \draw[domain=-2:2] plot(\x, \x+1);
\end{tikzpicture}
```

Example 12

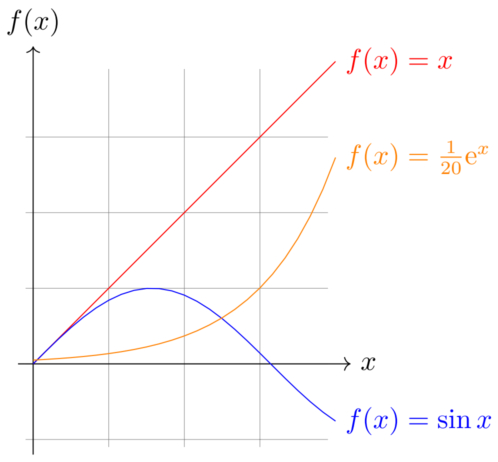


图 1.31: 12 TikZ Built-in Plot

```
\begin{tikzpicture}[domain=0:4]
    \draw[very thin,color=gray] (-0.1,-1.1) grid (3.9,3.9);
    \draw[->] (-0.2,0) -- (4.2,0) node[right] {$x$};
    \draw[->] (0,-1.2) -- (0,4.2) node[above] {$f(x)$};
    \draw[color=red] plot[id=x] function{x} node[right] {$f(x) =x$};
    \draw[color=blue] plot[id=sin] function{sin(x)} node[right] {$f(x) = \sin x$};
    \draw[color=orange] plot[id=exp] function{0.05*exp(x)} node[right] {$f(x) =
    \frac{1}{20} \mathrm e^x$};
\end{tikzpicture}
```

Example 13

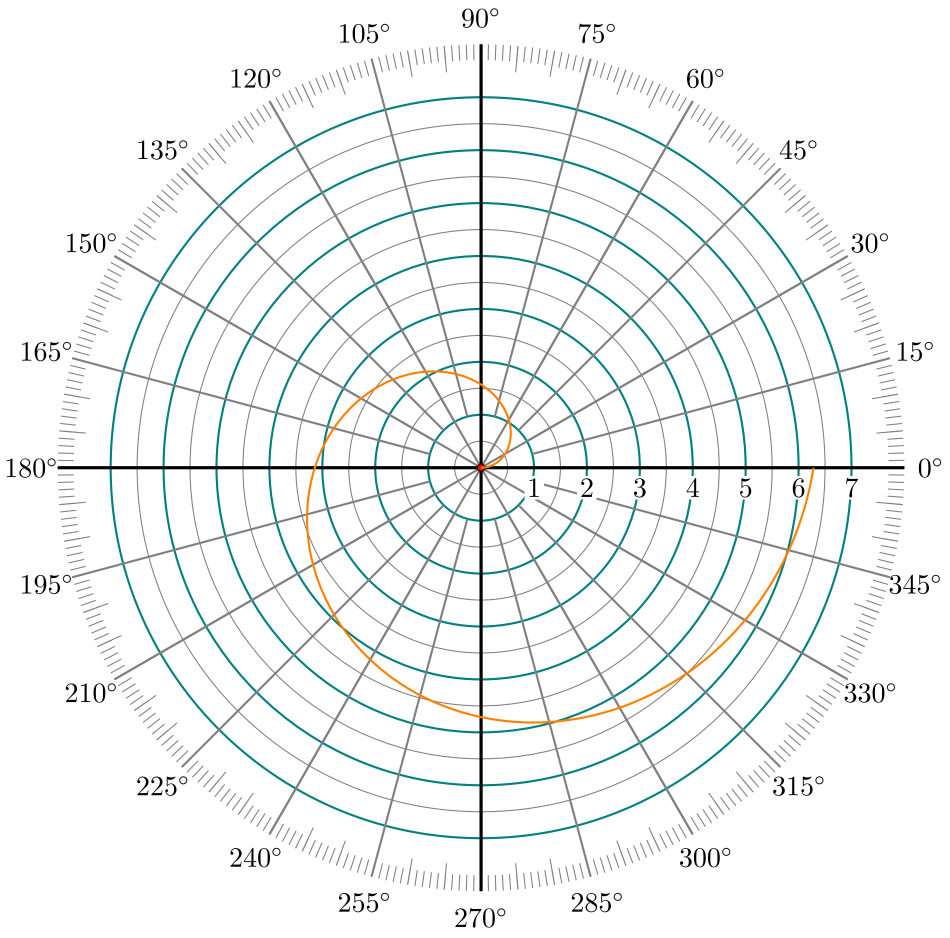


图 1.32: 13 Polar Plot And Coordinates

```
% https://texample.net/tikz/examples/polar-coordinates-template/
\begin{tikzpicture}[scale=.7]
    %Circles
    \foreach \r in {1, 2,...,7}
        \draw[teal, thick] (0,0) circle (\r);
    \foreach \r in {0.5, 1.5,...,7}
        \draw[gray, thin] (0,0) circle (\r);
    %1° Rays
    \foreach \a in {0, 1,...,359}
        \draw[gray] (\a:7.7) -- (\a:8);
    %5° Rays
    \foreach \a in {0, 5,...,359}
        \draw[gray] (\a:7.5) -- (\a:8);
    %15° Rays
    \foreach \a in {0, 15,...,359}
        \draw[thick,gray] (\a:1) -- (\a:8);
    %30° Rays
    \foreach \a in {0, 30,...,359}
        \draw[thick,gray] (0, 0) -- (\a:8);
    %Radius labels (background filled white)
    \foreach \r in {1, 2,...,7}
        \draw (\r,0) node[inner sep=1pt,below=3pt,rectangle,fill=white] {$\r$};
    %Main rays
    \foreach \a in {0, 90,...,359}
        \draw[very thick] (0, 0) -- (\a:8);
    %Angle labels
    \foreach \a in {0, 15,...,359}
        \draw (\a: 8.5) node {$\a^\circ$};
    %Central point
    \draw[fill=red] (0,0) circle(0.7mm);
    \PolarPlot[0:2*pi][thick, orange]{t}
\end{tikzpicture}
```

1.12.2 Mathematica

Example 1

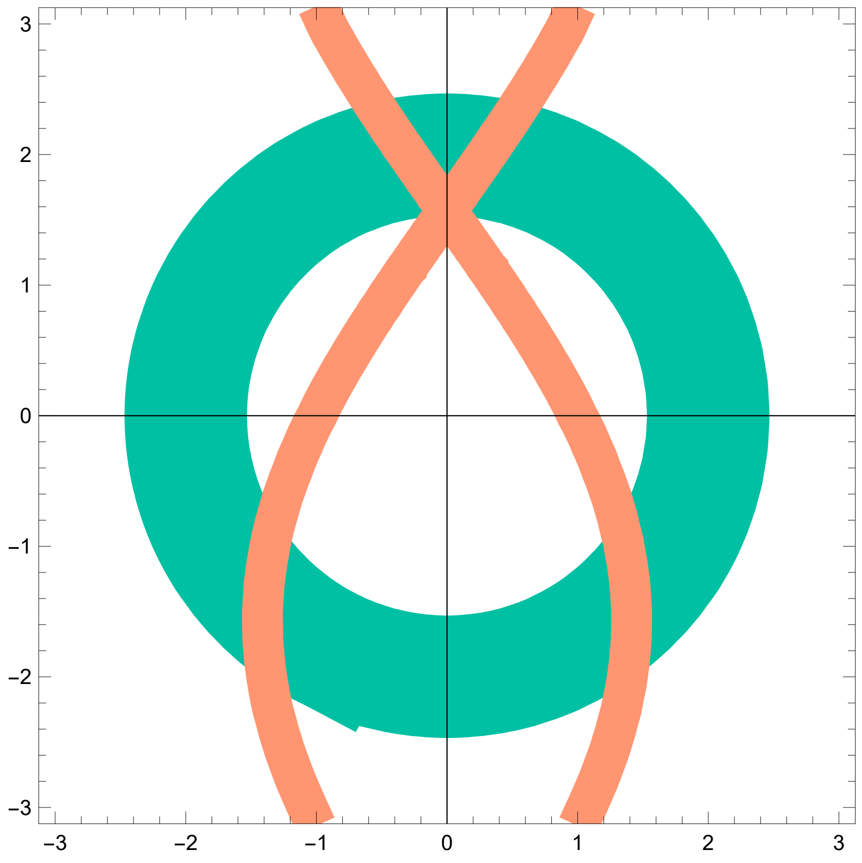


图 1.33: 1 Plot Stoke

```
\begin{wolframGraphics}[width=.75\linewidth]{example_1.wls}
fp1 = ContourPlot[x^2 + y^2 == 4, {x, -1.3, 0.6}, {y, -2.4, 3.2},
AspectRatio->(2.4+3.2)/(1.3+0.6), ContourStyle->Red];
fp2 = ContourPlot[x^2 + y^2 == 4, {x, -3, 3}, {y, -3, 3}, AspectRatio->1,
ContourStyle->RGBColor["#00C0A3"], AxesOrigin->{0, 0}, Axes->True];

fp3 = ContourPlot[{x^2 + y^2 == 4, x^2 + Sin[y] == 1},
    {x, -2.5, 2.5}, {y, -3, 3},
    ContourStyle->{
        {RGBColor["#00C0A3"], Thickness[0.15]},
        {RGBColor["#FF9671"], Thickness[0.05]}
    },
    AspectRatio->(3+3)/(2.5+2.5),
    AxesOrigin->{0,0},
    Axes->True,
    Frame->False,
    AxesStyle->Arrowheads[{0,0.03}]
]
figure = Show[fp2, fp1, fp3];
Export["./ztikz_output/mma_data/example_1.wls.pdf", figure];
\end{wolframGraphics}
```

Example 2

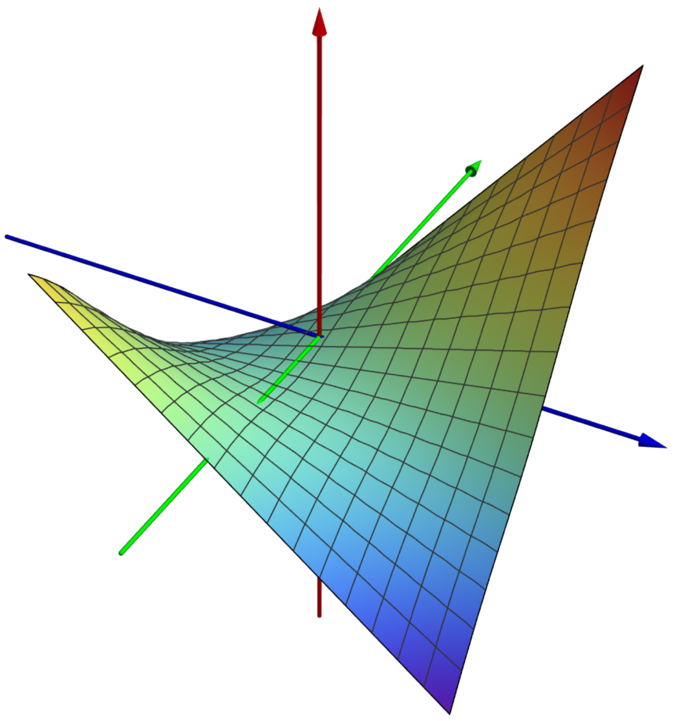


图 1.34: 2 3D Plot

```
\begin{wolframGraphics}[width=.75\linewidth]{example_2.wls}
(* 1. 定义一个产生箭头的命令 *)
arrow[start_, end_, type_] :=
Graphics3D[
    {type,
        { Arrowheads[.02], Arrow[Tube[{start, end}, 0.06]] }
    },
Boxed->False];

(* 2. 创建三个坐标轴的箭头，使用颜色进行区分 *)
xaxis = arrow[{-10, 0, 0}, {10, 0, 0}, Blue];
yaxis = arrow[{0, -10, 0}, {0, 10, 0}, Green];
zaxis = arrow[{0, 0, -10}, {0, 0, 10}, Red];

(* 3. 展示在同意坐标轴 *)
```

```
axis = {xaxis, yaxis, zaxis};

(* 4. 绘制一个函数由于测试 *)
fp4 = Plot3D[0.4*x + 0.2*Sin[y] + 0.2*x*y, {x, -5, 7}, {y, -6, 4},
ColorFunction->"Rainbow"];

(* 5. 显示三维函数图像和坐标轴 *)
figure = Show[axis, fp4]
Export["./ztikz_output/mma_data/example_2.wls.pdf", figure];
\end{wolframGraphics}
```

Example 3

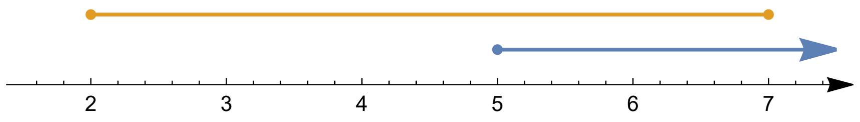


图 1.35: 3 Interval Plot

```
\begin{wolframGraphics}[width=.75\linewidth]{example_3.wls}
figure = NumberLinePlot[
    {Interval[{5, Infinity}], Interval[{2, 7}]},
    AxesStyle->Arrowheads[{0, 0.03}]
];
Export["./ztikz_output/mma_data/example_3.wls.pdf", figure];
\end{wolframGraphics}
```

Example 4

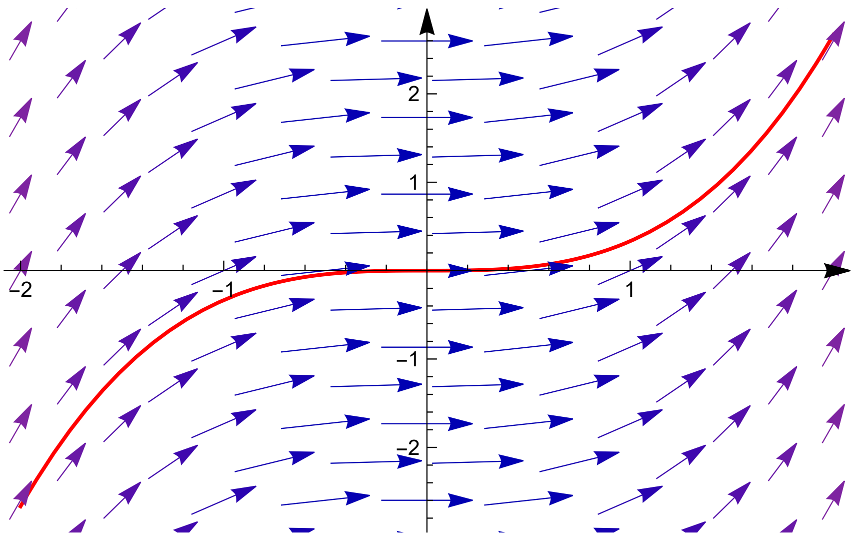


图 1.36: 4 Stream Plot

```
\begin{wolframGraphics}[width=.75\linewidth]{example_4.wls}
fvec = VectorPlot[{1, x^2}, {x, -4, 4}, {y, -4, 4}, AxesOrigin->{0, 0}, Axes->False,
Frame->False];
fp6 = Plot[1/3*x^3, {x, -2, 2}, PlotStyle->Red,  AxesStyle->Arrowheads[{0, 0.03}] ];
figure = Show[fp6, fvec];
Export["./ztikz_output/mma_data/example_4.wls.pdf", figure];
\end{wolframGraphics}
```

Example 5

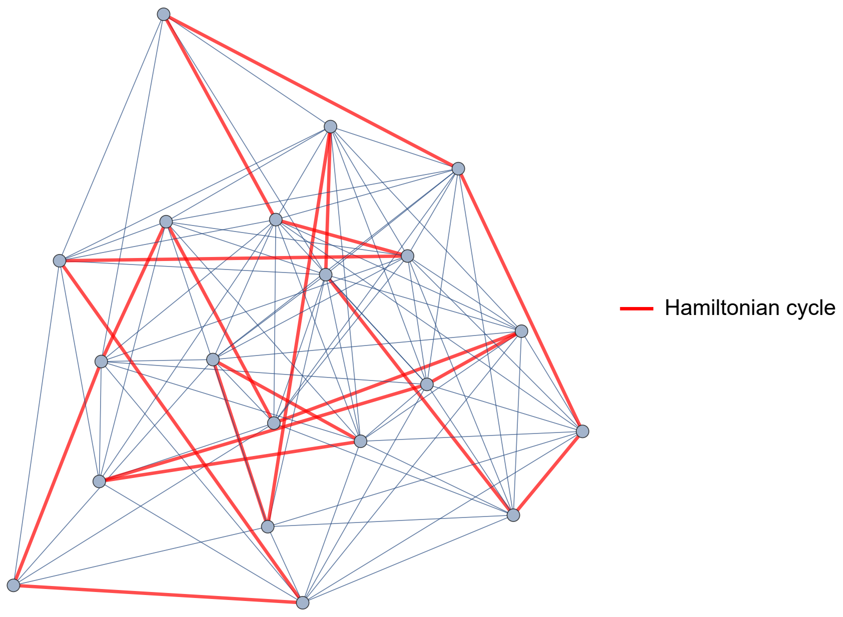


图 1.37: 5 Hamiltonian Graph

```
\begin{wolframGraphics}[width=.75\linewidth]{example_5.wls}
g=RandomGraph[{20,100}];
h=FindHamiltonianCycle[g];
figure = Legended[HighlightGraph[g,Style[h,Directive[Thick,Red]]],LineLegend[{Directive[
Thick,Red]},{"Hamiltonian
cycle"}]];
Export["./ztikz_output/mma_data/example_5.wls.pdf", figure];
\end{wolframGraphics}
```

Example 6

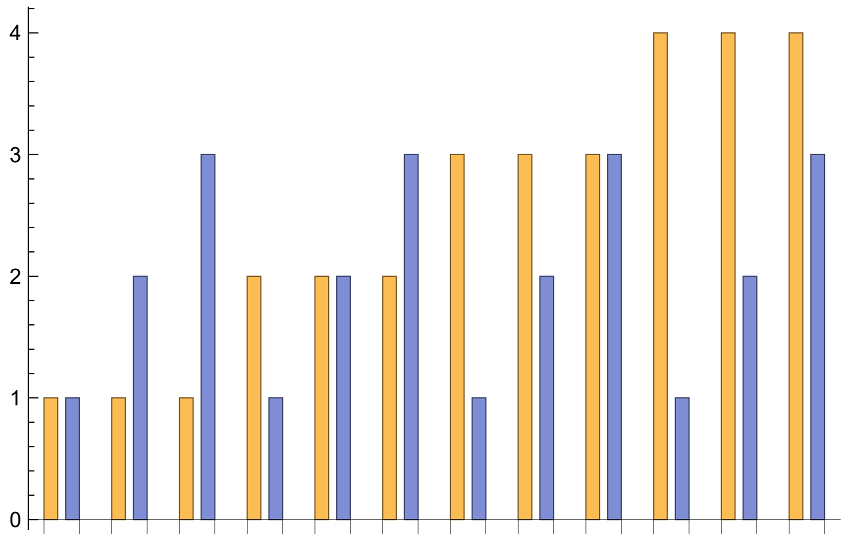


图 1.38: 6 statistic Plot

```
\begin{wolframGraphics}[width=.75\linewidth]{example_6.wls}
figure = BarChart[Flatten[Table[{i, j}, {i, 1, 4}, {j, 1, 3}], 1]];
Export["./ztikz_output/mma_data/example_6.wls.pdf", figure];
\end{wolframGraphics}
```

Example 7

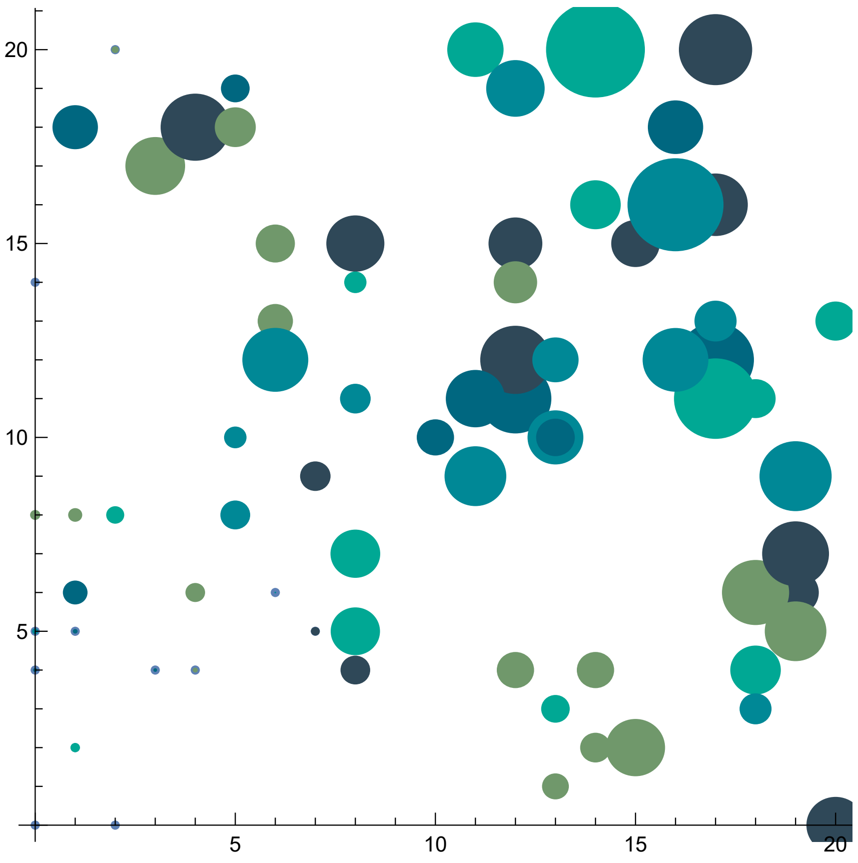


图 1.39: 7 Stream Plot

```
\begin{wolframGraphics}[width=.75\linewidth]{example_7.wls}
num = 80;
xls = RandomInteger[{0, 20}, num];
yls = RandomInteger[{0, 20}, num];

xycoor = {xls, yls}//Transpose;
color = {RGBColor["#00A894"], RGBColor["#008896"], RGBColor["#006780"],
RGBColor["#2F4858"], RGBColor["#70986B"]};

disks = Table[
    Graphics[{color[[RandomInteger[{1, 5}]]], Disk[xycoor[[i]], RandomReal[{0,
    0.05}]*#1+RandomReal[{0, 0.05}]*#2&[xycoor[[i]][[1]], xycoor[[i]][[2]]]]}],
    {i, 1, num}
];
fp91 = disks;
fp92 = ListPlot[xycoor, AspectRatio->(Max[yls])/(Max[xls])];
figure = Show[fp92, fp91];
```

```
Export["./ztikz_output/mma_data/example_7.wls.pdf", figure];
\end{wolframGraphics}
```

Example 8

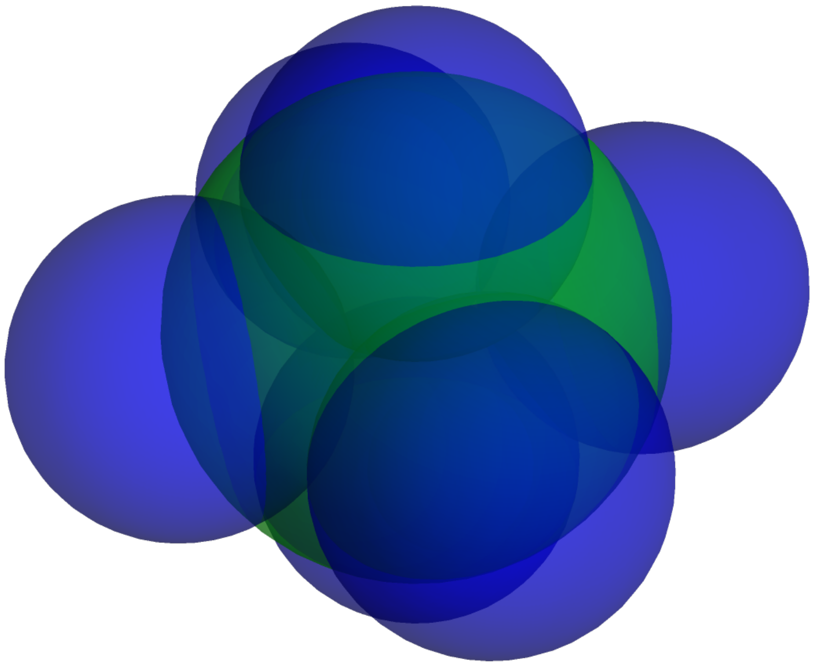


图 1.40: 8 3D Sphere

```
\begin{wolframGraphics}[width=.75\linewidth]{example_8.wls}
figure = Graphics3D[{
    Blue, Opacity[0.5], Sphere[{0.5, 0.5, 0}, 0.5],
    Blue, Opacity[0.5], Sphere[{-0.5, -0.5, 0}, 0.5],
    Blue, Opacity[0.5], Sphere[{0.5, -0.5, 0}, 0.5],
    Blue, Opacity[0.5], Sphere[{-0.5, 0.5, 0}, 0.5],
    Blue, Opacity[0.5], Sphere[{0, 0, 0.5}, 0.5],
    Blue, Opacity[0.5], Sphere[{0, 0, -0.5}, 0.5],
    Green,Sphere[{0,0,0}, 0.75]
    }, Boxed->False
];
Export["./ztikz_output/mma_data/example_8.wls.pdf", figure];
\end{wolframGraphics}
```

1.12.3 Python

Example 1

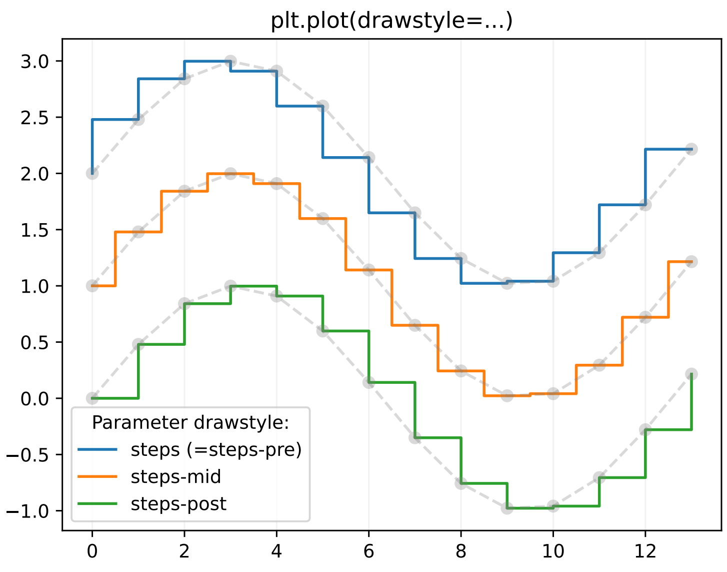


图 1.41: 1 Step Demo

```
\begin{pyfig}[width=.75\linewidth]{example_1.mpl}
# https://matplotlib.org/stable/gallery/lines_bars_and_markers/step_demo.html

import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(14)
y = np.sin(x / 2)

plt.plot(x, y + 2, drawstyle='steps', label='steps (=steps-pre)')
plt.plot(x, y + 2, 'o--', color='grey', alpha=0.3)

plt.plot(x, y + 1, drawstyle='steps-mid', label='steps-mid')
plt.plot(x, y + 1, 'o--', color='grey', alpha=0.3)

plt.plot(x, y, drawstyle='steps-post', label='steps-post')
plt.plot(x, y, 'o--', color='grey', alpha=0.3)
plt.grid(axis='x', color='0.95')
plt.legend(title='Parameter drawstyle:')
plt.title('plt.plot(drawstyle=...)')
\end{pyfig}
```

Example 2

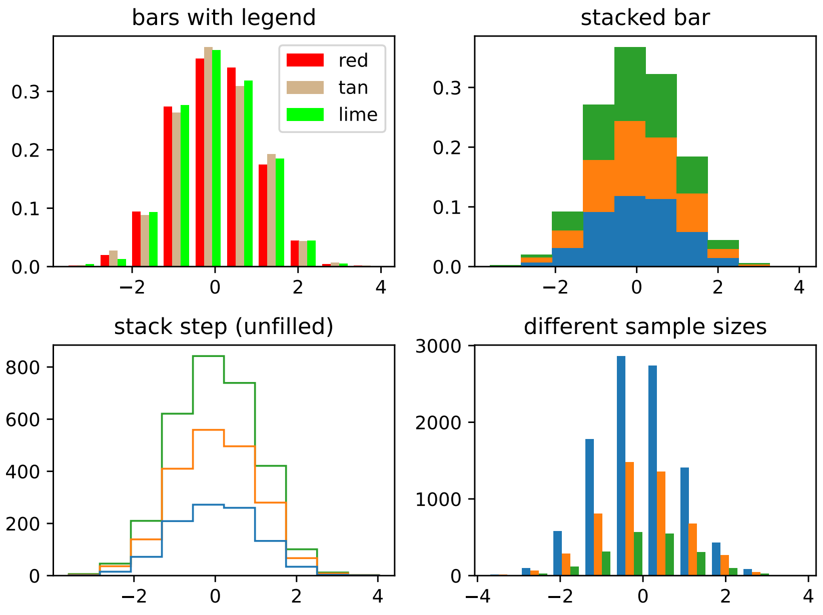


图 1.42: 2 Histogram Demo

```
\begin{pyfig}[width=.75\linewidth]{example_2.mpl}
# https://matplotlib.org/stable/gallery/lines_bars_and_markers/histogram_demo.html

import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
import numpy as np

np.random.seed(19680801)

n_bins = 10
x = np.random.randn(1000, 3)

fig, ((ax0, ax1), (ax2, ax3)) = plt.subplots(nrows=2, ncols=2)

colors = ['red', 'tan', 'lime']
ax0.hist(x, n_bins, density=True, histtype='bar', color=colors, label=colors)
ax0.legend(prop={'size': 10})
ax0.set_title('bars with legend')

ax1.hist(x, n_bins, density=True, histtype='bar', stacked=True)
ax1.set_title('stacked bar')
```

```
ax2.hist(x, n_bins, histtype='step', stacked=True, fill=False)
ax2.set_title('stack step (unfilled)')

# Make a multiple-histogram of data-sets with different length.
x_multi = [np.random.randn(n) for n in [10000, 5000, 2000]]
ax3.hist(x_multi, n_bins, histtype='bar')
ax3.set_title('different sample sizes')

fig.tight_layout()
\end{pyfig}
```

Example 3

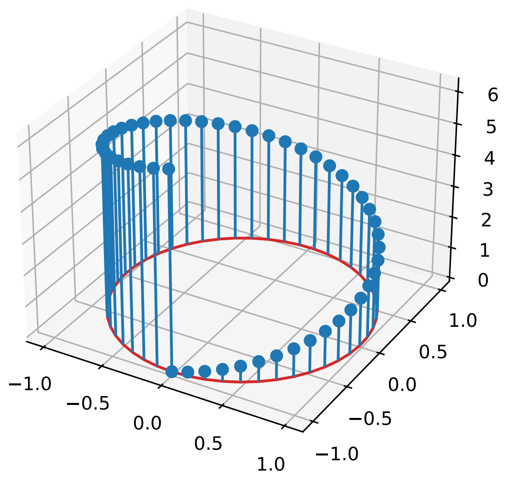


图 1.43: 3 3D Stem Demo

```
\begin{pyfig}[width=.75\linewidth]{example_3.mpl}
# https://matplotlib.org/stable/gallery/mplot3d/stem3d_demo.html

import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
import numpy as np

theta = np.linspace(0, 2*np.pi)
x = np.cos(theta - np.pi/2)
y = np.sin(theta - np.pi/2)
z = theta

fig, ax = plt.subplots(subplot_kw=dict(projection='3d'))
ax.stem(x, y, z)
\end{pyfig}
```

Example 4

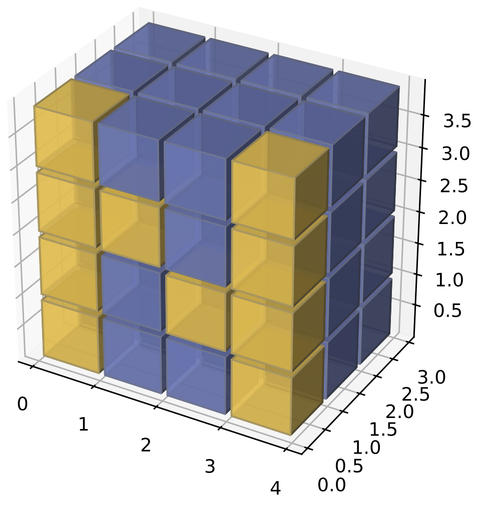


图 1.44: 4 3D Numpy Logo

```
\begin{pyfig}[width=.75\linewidth]{example_4.mpl}
# https://matplotlib.org/stable/gallery/mplot3d/voxels_numpy_logo.html
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
import numpy as np


def explode(data):
    size = np.array(data.shape)*2
    data_e = np.zeros(size - 1, dtype=data.dtype)
    data_e[::2, ::2, ::2] = data
    return data_e

# build up the numpy logo
n_voxels = np.zeros((4, 3, 4), dtype=bool)
n_voxels[0, 0, :] = True
n_voxels[-1, 0, :] = True
n_voxels[1, 0, 2] = True
n_voxels[2, 0, 1] = True
facecolors = np.where(n_voxels, '#FFD65DC0', '#7A88CCC0')
edgecolors = np.where(n_voxels, '#BFAB6E', '#7D84A6')
```

```
filled = np.ones(n_voxels.shape)

# upscale the above voxel image, leaving gaps
filled_2 = explode(filled)
fcolors_2 = explode(facecolors)
ecolors_2 = explode(edgecolors)

# Shrink the gaps
x, y, z = np.indices(np.array(filled_2.shape) + 1).astype(float) // 2
x[0::2, :, :] += 0.05
y[:, 0::2, :] += 0.05
z[:, :, 0::2] += 0.05
x[1::2, :, :] += 0.95
y[:, 1::2, :] += 0.95
z[:, :, 1::2] += 0.95

ax = plt.figure().add_subplot(projection='3d')
ax.voxels(x, y, z, filled_2, facecolors=fcolors_2, edgecolors=ecolors_2)
ax.set_aspect('equal')
\end{pyfig}
```

第二部分

索引与参考

CHAPTER 2

部分命令/名词索引

CHAPTER 3

TikZ Built-in Functions

本节主要截取了 TikZ 官方文档中的部分内容，主要涵盖 TikZ 内置函数的绘制的内容。相信这一部分的内容可以帮助读者更好的理解 zTikZ 宏包的工作原理，同时也能够使得 zTikZ 宏包的使用更加的便捷.

22 Plots of Functions

A warning before we get started: *If you are looking for an easy way to create a normal plot of a function with scientific axes, ignore this section and instead look at the `pgfplots` package or at the `datavisualization` command from Part VI.*

22.1 Overview

Ti*k*Z can be used to create plots of functions, a job that is normally handled by powerful programs like GNUPLOT or MATHEMATICA. These programs can produce two different kinds of output: First, they can output a complete plot picture in a certain format (like PDF) that includes all low-level commands necessary for drawing the complete plot (including axes and labels). Second, they can usually also produce "just plain data" in the form of a long list of coordinates. Most of the powerful programs consider it a to be "a bit boring" to just output tabled data and very much prefer to produce fancy pictures. Nevertheless, when coaxed, they can also provide the plain data.

The advantage of creating plots directly using Ti*k*Z is *consistency:* Plots created using Ti*k*Z will automatically have the same styling and fonts as those used in the rest of a document – something that is hard to do right when an external program gets involved. Other problems people encounter with external programs include: Formulas will look different, if they can be rendered at all; line widths will usually be too thick or too thin; scaling effects upon inclusion can create a mismatch between sizes in the plot and sizes in the text; the automatic grid generated by most programs is mostly distracting; the automatic ticks generated by most programs are cryptic numerics (try adding a tick reading "π" at the right point); most programs make it very easy to create "chart junk" in a most convenient fashion; arrows and plot marks will almost never match the arrows used in the rest of the document. This list is not exhaustive, unfortunately.

There are basically three ways of creating plots using Ti*k*Z:

1. Use the `plot` path operation. How this works is explained in the present section. This is the most "basic" of the three options and forces you to do a lot of things "by hand" like adding axes or ticks.

2. Use the `datavisualization` path command, which is documented in Part VI. This command is much more powerful than the `plot` path operation and produces complete plots including axes and ticks. The downside is that you cannot use it to "just" quickly plot a simple curve (or, more precisely, it is hard to use it in this way).

3. Use the `pgfplots` package, which is basically an alternative to the `datavisualization` command. While the underlying philosophy of this package is not as "ambitious" as that of the command `datavisualization`, it is somewhat more mature, has a simpler design, and wider support base.

22.2 The Plot Path Operation

The `plot` path operation can be used to append a line or curve to the path that goes through a large number of coordinates. These coordinates are either given in a simple list of coordinates, read from some file, or they are computed on the fly.

The syntax of the `plot` comes in different versions.

`\path` … `--plot`⟨*further arguments*⟩ …;

This operation plots the curve through the coordinates specified in the ⟨*further arguments*⟩. The current (sub)path is simply continued, that is, a line-to operation to the first point of the curve is implicitly added. The details of the ⟨*further arguments*⟩ will be explained in a moment.

`\path` … `plot`⟨*further arguments*⟩ …;

This operation plots the curve through the coordinates specified in the ⟨*further arguments*⟩ by first "moving" to the first coordinate of the curve.

The ⟨*further arguments*⟩ are used in different ways to specifying the coordinates of the points to be plotted:

1. `--plot[`⟨*local options*⟩`]coordinates{`⟨*coordinate 1*⟩⟨*coordinate 2*⟩…⟨*coordinate n*⟩`}`

2. `--plot[`⟨*local options*⟩`]file{`⟨*filename*⟩`}`

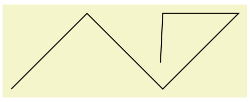
3. `--plot[`⟨*local options*⟩`]`⟨*coordinate expression*⟩

4. `--plot[`⟨*local options*⟩`]function{`⟨*gnuplot formula*⟩`}`

These different ways are explained in the following.

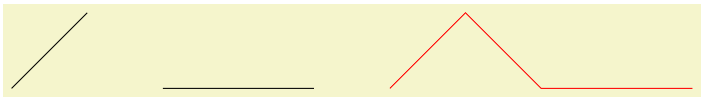
22.3 Plotting Points Given Inline

Points can be given directly in the TEX-file as in the following example:



```
\tikz \draw plot coordinates {(0,0) (1,1) (2,0) (3,1) (2,1) (10:2cm)};
```

Here is an example showing the difference between `plot` and `--plot`:



```
\begin{tikzpicture}
  \draw (0,0) -- (1,1) plot coordinates {(2,0)  (4,0)};
  \draw[color=red,xshift=5cm]
        (0,0) -- (1,1) -- plot coordinates {(2,0)  (4,0)};
\end{tikzpicture}
```

22.4 Plotting Points Read From an External File

The second way of specifying points is to put them in an external file named ⟨*filename*⟩. Currently, the only file format that Ti*k*Z allows is the following: Each line of the ⟨*filename*⟩ should contain one line starting with two numbers, separated by a space. A line may also be empty or, if it starts with `#` or `%` it is considered empty. For such lines, a "new data set" is started, typically resulting in a new subpath being started in the plot (see Section 112.2.2 on how to change this behavior, if necessary). For lines containing two numbers, they must be separated by a space. They may be following by arbitrary text, which is ignored, *except* if it is `o` or `u`. In the first case, the point is considered to be an *outlier* and normally also results in a new subpath being started. In the second case, the point is considered to be *undefined*, which also results in a new subpath being started. Again, see Section 112.2.2 on how to change this, if necessary. (This is exactly the format that GNUPLOT produces when you say `set table`.)

```
\tikz \draw plot[mark=x,smooth] file {plots/pgfmanual-sine.table};
```

The file `plots/pgfmanual-sine.table` reads:

```
#Curve 0, 20 points
#x y type
0.00000 0.00000  i
0.52632 0.50235  i
1.05263 0.86873  i
1.57895 0.99997  i
...
9.47368 -0.04889  i
10.00000 -0.54402  i
```

It was produced from the following source, using `gnuplot`:

```
set table  "../plots/pgfmanual-sine.table"
set format "%.5f"
set samples 20
plot [x=0:10] sin(x)
```

The ⟨*local options*⟩ of the `plot` operation are local to each plot and do not affect other plots "on the same path". For example, `plot[yshift=1cm]` will locally shift the plot 1cm upward. Remember, however, that most options can only be applied to paths as a whole. For example, `plot[red]` does not have the effect of making the plot red. After all, you are trying to "locally" make part of the path red, which is not possible.

22.5 Plotting a Function

When you plot a function, the coordinates of the plot data can be computed by evaluating a mathematical expression. Since PGF comes with a mathematical engine, you can specify this expression and then have Ti*k*Z produce the desired coordinates for you, automatically.

Since this case is quite common when plotting a function, the syntax is easy: Following the `plot` command and its local options, you directly provide a ⟨*coordinate expression*⟩. It looks like a normal coordinate, but inside you may use a special macro, which is `\x` by default, but this can be changed using the `variable` option. The ⟨*coordinate expression*⟩ is then evaluated for different values for `\x` and the resulting coordinates are plotted.

Note that you will often have to put the x- or y-coordinate inside braces, namely whenever you use an expression involving a parenthesis.

The following options influence how the ⟨*coordinate expression*⟩ is evaluated:

`/tikz/variable=`⟨*macro*⟩ (no default, initially `\x`)

Sets the macro whose value is set to the different values when ⟨*coordinate expression*⟩ is evaluated.

`/tikz/samples=`⟨*number*⟩ (no default, initially `25`)

Sets the number of samples used in the plot.

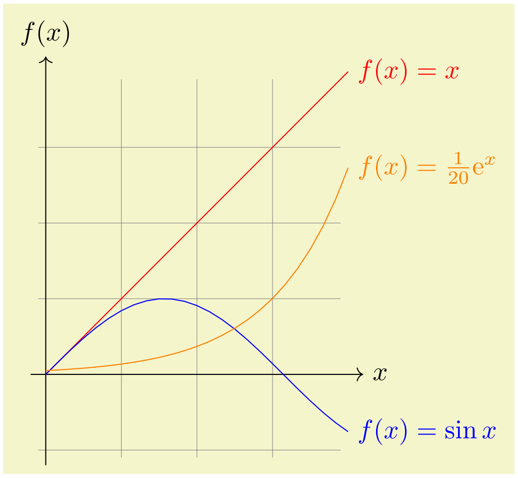
`/tikz/domain=`⟨*start*⟩`:`⟨*end*⟩ (no default, initially `-5:5`)

Sets the domain from which the samples are taken.

`/tikz/samples at=`⟨*sample list*⟩ (no default)

This option specifies a list of positions for which the variable should be evaluated. For instance, you can say `samples at={1,2,8,9,10}` to have the variable evaluated exactly for values 1, 2, 8, 9, and 10. You can use the `\foreach` syntax, so you can use `...` inside the ⟨*sample list*⟩.

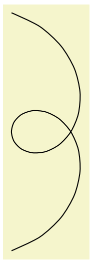
When this option is used, the `samples` and `domain` option are overruled. The other way round, setting either `samples` or `domain` will overrule this option.



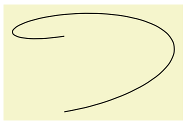
```
\begin{tikzpicture}[domain=0:4]
  \draw[very thin,color=gray] (-0.1,-1.1) grid (3.9,3.9);

  \draw[->] (-0.2,0) -- (4.2,0) node[right] {$x$};
  \draw[->] (0,-1.2) -- (0,4.2) node[above] {$f(x)$};

  \draw[color=red]    plot (\x,\x)             node[right] {$f(x) =x$};
  % \x r means to convert '\x' from degrees to _r_adians:
  \draw[color=blue]   plot (\x,{sin(\x r)})    node[right] {$f(x) = \sin x$};
  \draw[color=orange] plot (\x,{0.05*exp(\x)}) node[right] {$f(x) = \frac{1}{20} \mathrm e^x$};
\end{tikzpicture}
```



```
\tikz \draw[scale=0.5,domain=-3.141:3.141,smooth,variable=\t]
  plot ({\t*sin(\t r)},{\t*cos(\t r)});
```



```
\tikz \draw[domain=0:360,smooth,variable=\t]
  plot ({sin(\t)},\t/360,{cos(\t)});
```

22.6 Plotting a Function Using Gnuplot

Often, you will want to plot points that are given via a function like $f(x) = x \sin x$. Unfortunately, TEX does not really have enough computational power to generate the points of such a function efficiently (it is a text processing program, after all). However, if you allow it, TEX can try to call external programs that can easily produce the necessary points. Currently, Ti*k*Z knows how to call GNUPLOT.

When Ti*k*Z encounters your operation `plot[id=`⟨*id*⟩`] function{x*sin(x)}` for the first time, it will create a file called ⟨*prefix*⟩⟨*id*⟩`.gnuplot`, where ⟨*prefix*⟩ is `\jobname.` by default, that is, the name of your main `.tex` file. If no ⟨*id*⟩ is given, it will be empty, which is alright, but it is better when each plot has a unique ⟨*id*⟩ for reasons explained in a moment. Next, Ti*k*Z writes some initialization code into this file followed by `plot x*sin(x)`. The initialization code sets up things such that the `plot` operation will write the coordinates into another file called ⟨*prefix*⟩⟨*id*⟩`.table`. Finally, this table file is read as if you had said `plot file{`⟨*prefix*⟩⟨*id*⟩`.table}`.

For the plotting mechanism to work, two conditions must be met:

1. You must have allowed TEX to call external programs. This is often switched off by default since this is a security risk (you might, without knowing, run a TEX file that calls all sorts of "bad" commands). To enable this "calling external programs" a command line option must be given to the TEX program. Usually, it is called something like `shell-escape` or `enable-write18`. For example, for my `pdflatex` the option `--shell-escape` can be given.

2. You must have installed the `gnuplot` program and TEX must find it when compiling your file.

Unfortunately, these conditions will not always be met. Especially if you pass some source to a coauthor and the coauthor does not have GNUPLOT installed, he or she will have trouble compiling your files.

For this reason, Ti*k*Z behaves differently when you compile your graphic for the second time: If upon reaching `plot[id=`⟨*id*⟩`] function{...}` the file ⟨*prefix*⟩⟨*id*⟩`.table` already exists *and* if the ⟨*prefix*⟩⟨*id*⟩`.gnuplot` file contains what Ti*k*Z thinks that it "should" contain, the `.table` file is immediately read without trying to call a `gnuplot` program. This approach has the following advantages:

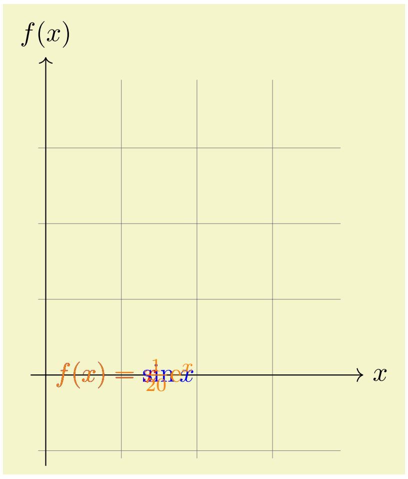
1. If you pass a bundle of your `.tex` file and all `.gnuplot` and `.table` files to someone else, that person can TEX the `.tex` file without having to have `gnuplot` installed.

2. If the `\write18` feature is switched off for security reasons (a good idea), then, upon the first compilation of the `.tex` file, the `.gnuplot` will still be generated, but not the `.table` file. You can then simply call `gnuplot` "by hand" for each `.gnuplot` file, which will produce all necessary `.table` files.

3. If you change the function that you wish to plot or its domain, Ti*k*Z will automatically try to regenerate the `.table` file.

4. If, out of laziness, you do not provide an `id`, the same `.gnuplot` will be used for different plots, but this is not a problem since the `.table` will automatically be regenerated for each plot on-the-fly. *Note: If you intend to share your files with someone else, always use an id, so that the file can by typeset without having* GNUPLOT *installed.* Also, having unique ids for each plot will improve compilation speed since no external programs need to be called, unless it is really necessary.

When you use `plot function{`⟨*gnuplot formula*⟩`}`, the ⟨*gnuplot formula*⟩ must be given in the `gnuplot` syntax, whose details are beyond the scope of this manual. Here is the ultra-condensed essence: Use `x` as the variable and use the C-syntax for normal plots, use `t` as the variable for parametric plots. Here are some examples:



```
\begin{tikzpicture}[domain=0:4]
  \draw[very thin,color=gray] (-0.1,-1.1) grid (3.9,3.9);

  \draw[->] (-0.2,0) -- (4.2,0) node[right] {$x$};
  \draw[->] (0,-1.2) -- (0,4.2) node[above] {$f(x)$};

  \draw[color=red]    plot[id=x]   function{x}           node[right] {$f(x) =x$};
  \draw[color=blue]   plot[id=sin] function{sin(x)}      node[right] {$f(x) = \sin x$};
  \draw[color=orange] plot[id=exp] function{0.05*exp(x)} node[right] {$f(x) = \frac{1}{20} \mathrm e^x$};
\end{tikzpicture}
```

The plot is influenced by the following options: First, the options `samples` and `domain` explained earlier. Second, there are some more specialized options.

`/tikz/parametric=`⟨*boolean*⟩ (default `true`)

Sets whether the plot is a parametric plot. If true, then `t` must be used instead of `x` as the parameter and two comma-separated functions must be given in the ⟨*gnuplot formula*⟩. An example is the following:

```
\tikz \draw[scale=0.5,domain=-3.141:3.141,smooth]
  plot[parametric,id=parametric-example] function{t*sin(t),t*cos(t)};
```

`/tikz/range=`⟨*start*⟩`:`⟨*end*⟩ (no default)

This key sets the range of the plot. If set, all points whose y-coordinates lie outside this range will be considered to be outliers and will cause jumps in the plot, by default:

```
\tikz \draw[scale=0.5,domain=-3.141:3.141, samples=100, smooth, range=-3:3]
  plot[id=tan-example] function{tan(x)};
```

`/tikz/yrange=`⟨*start*⟩`:`⟨*end*⟩ (no default)

Same as `range`.

`/tikz/xrange=`⟨*start*⟩`:`⟨*end*⟩ (no default)

Set the x-range. This makes sense only for parametric plots.

```
\tikz \draw[scale=0.5,domain=-3.141:3.141,smooth,xrange=0:1]
  plot[parametric,id=parametric-example-cut] function{t*sin(t),t*cos(t)};
```

`/tikz/id=`⟨*id*⟩ (no default)

Sets the identifier of the current plot. This should be a unique identifier for each plot (though things will also work if it is not, but not as well, see the explanations above). The ⟨*id*⟩ will be part of a filename, so it should not contain anything fancy like `*` or `$`.

`/tikz/prefix=`⟨*prefix*⟩ (no default)

The ⟨*prefix*⟩ is put before each plot file name. The default is `\jobname.`, but if you have many plots, it might be better to use, say `plots/` and have all plots placed in a directory. You have to create the directory yourself.

`/tikz/raw gnuplot` (no value)

This key causes the ⟨*gnuplot formula*⟩ to be passed on to GNUPLOT without setting up the samples or the `plot` operation. Thus, you could write

```
plot[raw gnuplot,id=raw-example] function{set samples 25; plot sin(x)}
```

This can be useful for complicated things that need to be passed to GNUPLOT. However, for really complicated situations you should create a special external generating GNUPLOT file and use the `file`-syntax to include the table “by hand”.

The following styles influence the plot:

`/tikz/every plot` (style, initially empty)

This style is installed in each plot, that is, as if you always said

```
  plot[every plot,...]
```

This is most useful for globally setting a prefix for all plots by saying:

```
\tikzset{every plot/.style={prefix=plots/}}
```

22.7 Placing Marks on the Plot

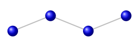
As we saw already, it is possible to add *marks* to a plot using the `mark` option. When this option is used, a copy of the plot mark is placed on each point of the plot. Note that the marks are placed *after* the whole path has been drawn/filled/shaded. In this respect, they are handled like text nodes.

In detail, the following options govern how marks are drawn:

`/tikz/mark=`⟨*mark mnemonic*⟩ (no default)

Sets the mark to a mnemonic that has previously been defined using the `\pgfdeclareplotmark`. By default, `*`, `+`, and `x` are available, which draw a filled circle, a plus, and a cross as marks. Many more marks become available when the library `plotmarks` is loaded. Section 65.6 lists the available plot marks.

One plot mark is special: the `ball` plot mark is available only in Ti*k*Z. The `ball color` option determines the balls’s color. Do not use this option with a large number of marks since it will take very long to render in PostScript.

Option	Effect
`mark=ball`	

`/tikz/mark repeat=`⟨*r*⟩ (no default)

This option tells Ti*k*Z that only every rth mark should be drawn.

```
\tikz \draw plot[mark=x,mark repeat=3,smooth] file {plots/pgfmanual-
sine.table};
```

`/tikz/mark phase=`⟨*p*⟩ (no default)

This option tells Ti*k*Z that the first mark to be draw should be the pth, followed by the $(p+r)$th, then the $(p+2r)$th, and so on.

```
\tikz \draw plot[mark=x,mark repeat=3,mark phase=6,smooth] file {plots/pgfmanual-
sine.table};
```

`/tikz/mark indices=`⟨*list*⟩ (no default)

This option allows you to specify explicitly the indices at which a mark should be placed. Counting starts with 1. You can use the `\foreach` syntax, that is, `...` can be used.

```
\tikz \draw plot[mark=x,mark indices={1,4,...,10,11,12,...,16,20},smooth]
  file {plots/pgfmanual-sine.table};
```

`/tikz/mark size=`⟨*dimension*⟩ (no default)

Sets the size of the plot marks. For circular plot marks, ⟨*dimension*⟩ is the radius, for other plot marks ⟨*dimension*⟩ should be about half the width and height.

This option is not really necessary, since you achieve the same effect by specifying `scale=`⟨*factor*⟩ as a local option, where ⟨*factor*⟩ is the quotient of the desired size and the default size. However, using `mark size` is a bit faster and more natural.

`/tikz/every mark` (style, no value)

This style is installed before drawing plot marks. For example, you can scale (or otherwise transform) the plot mark or set its color.

`/tikz/mark options=`⟨*options*⟩ (no default)

Redefines `every mark` such that it sets `{`⟨*options*⟩`}`.

```
\tikz \fill[fill=blue!20]
  plot[mark=triangle*,mark options={color=blue,rotate=180}]
    file{plots/pgfmanual-sine.table} |- (0,0);
```

`/tikz/no marks` (style, no value)

Disables markers (the same as `mark=none`).

`/tikz/no markers` (style, no value)

Disables markers (the same as `mark=none`).

22.8 Smooth Plots, Sharp Plots, Jump Plots, Comb Plots and Bar Plots

There are different things the `plot` operation can do with the points it reads from a file or from the inlined list of points. By default, it will connect these points by straight lines. However, you can also use options to change the behavior of `plot`.

`/tikz/sharp plot` (no value)

This is the default and causes the points to be connected by straight lines. This option is included only so that you can "switch back" if you "globally" install, say, `smooth`.

`/tikz/smooth` (no value)

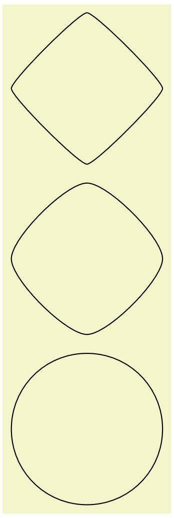
This option causes the points on the path to be connected using a smooth curve:

```
\tikz\draw plot[smooth] file{plots/pgfmanual-sine.table};
```

Note that the smoothing algorithm is not very intelligent. You will get the best results if the bending angles are small, that is, less than about 30° and, even more importantly, if the distances between points are about the same all over the plotting path.

`/tikz/tension=`⟨*value*⟩ (no default)

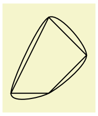
This option influences how "tight" the smoothing is. A lower value will result in sharper corners, a higher value in more "round" curves. A value of 1 results in a circle if four points at quarter-positions on a circle are given. The default is 0.55. The "correct" value depends on the details of plot.



```
\begin{tikzpicture}[smooth cycle]
  \draw                 plot[tension=0.2]
    coordinates{(0,0) (1,1) (2,0) (1,-1)};
  \draw[yshift=-2.25cm] plot[tension=0.5]
    coordinates{(0,0) (1,1) (2,0) (1,-1)};
  \draw[yshift=-4.5cm]  plot[tension=1]
    coordinates{(0,0) (1,1) (2,0) (1,-1)};
\end{tikzpicture}
```

/tikz/smooth cycle (no value)

This option causes the points on the path to be connected using a closed smooth curve.



```
\tikz[scale=0.5]
  \draw plot[smooth cycle] coordinates{(0,0) (1,0) (2,1) (1,2)}
        plot               coordinates{(0,0) (1,0) (2,1) (1,2)} -- cycle;
```

/tikz/const plot (no value)

This option causes the points on the path to be connected using piecewise constant series of lines:

```
\tikz\draw plot[const plot] file{plots/pgfmanual-sine.table};
```

/tikz/const plot mark left (no value)

Just an alias for `/tikz/const plot`.

```
\tikz\draw plot[const plot mark left,mark=*] file{plots/pgfmanual-sine.table};
```

/tikz/const plot mark right (no value)

A variant of `/tikz/const plot` which places its mark on the right ends:

```
\tikz\draw plot[const plot mark right,mark=*] file{plots/pgfmanual-sine.table};
```

/tikz/const plot mark mid (no value)

A variant of `/tikz/const plot` which places its mark in the middle of the horizontal lines:

```
\tikz\draw plot[const plot mark mid,mark=*] file{plots/pgfmanual-sine.table};
```

More precisely, it generates vertical lines in the middle between each pair of consecutive points. If the mesh width is constant, this leads to symmetrically placed marks (“middle”).

/tikz/jump mark left (no value)

This option causes the points on the path to be drawn using piecewise constant, non-connected series of lines. If there are any marks, they will be placed on left open ends:

```
\tikz\draw plot[jump mark left, mark=*] file{plots/pgfmanual-sine.table};
```

`/tikz/jump mark right` (no value)

This option causes the points on the path to be drawn using piecewise constant, non-connected series of lines. If there are any marks, they will be placed on right open ends:

```
\tikz\draw plot[jump mark right, mark=*] file{plots/pgfmanual-sine.table};
```

`/tikz/jump mark mid` (no value)

This option causes the points on the path to be drawn using piecewise constant, non-connected series of lines. If there are any marks, they will be placed in the middle of the horizontal line segments:

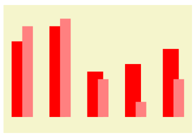
```
\tikz\draw plot[jump mark mid, mark=*] file{plots/pgfmanual-sine.table};
```

In case of non-constant mesh widths, the same remarks as for `const plot mark mid` apply.

`/tikz/ycomb` (no value)

This option causes the `plot` operation to interpret the plotting points differently. Instead of connecting them, for each point of the plot a straight line is added to the path from the x-axis to the point, resulting in a sort of "comb" or "bar diagram".

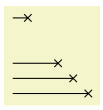
```
\tikz\draw[ultra thick] plot[ycomb,thin,mark=*] file{plots/pgfmanual-
sine.table};
```



```
\begin{tikzpicture}[ycomb]
  \draw[color=red,line width=6pt]
    plot coordinates{(0,1) (.5,1.2) (1,.6) (1.5,.7) (2,.9)};
  \draw[color=red!50,line width=4pt,xshift=3pt]
    plot coordinates{(0,1.2) (.5,1.3) (1,.5) (1.5,.2) (2,.5)};
\end{tikzpicture}
```

`/tikz/xcomb` (no value)

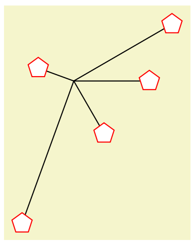
This option works like `ycomb` except that the bars are horizontal.



```
\tikz \draw plot[xcomb,mark=x] coordinates{(1,0) (0.8,0.2) (0.6,0.4) (0.2,1)};
```

`/tikz/polar comb` (no value)

This option causes a line from the origin to the point to be added to the path for each plot point.

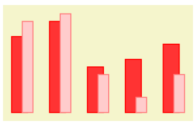


```
\tikz \draw plot[polar comb,
     mark=pentagon*,mark options={fill=white,draw=red},mark size=4pt]
   coordinates {(0:1cm) (30:1.5cm) (160:.5cm) (250:2cm) (-60:.8cm)};
```

`/tikz/ybar` (no value)

This option produces fillable bar plots. It is thus very similar to `ycomb`, but it employs rectangular shapes instead of line-to operations. It thus allows to use any fill or pattern style.

```
\tikz\draw[draw=blue,fill=blue!60!black] plot[ybar] file{plots/pgfmanual-
sine.table};
```

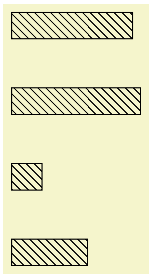


```
\begin{tikzpicture}[ybar]
  \draw[color=red,fill=red!80,bar width=6pt]
    plot coordinates{(0,1) (.5,1.2) (1,.6) (1.5,.7) (2,.9)};
  \draw[color=red!50,fill=red!20,bar width=4pt,bar shift=3pt]
    plot coordinates{(0,1.2) (.5,1.3) (1,.5) (1.5,.2) (2,.5)};
\end{tikzpicture}
```

The use of `bar width` and `bar shift` is explained in the `plothandlers` library documentation, section 65.4. Please refer to page 753.

`/tikz/xbar` (no value)

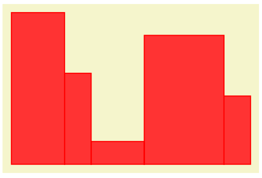
This option works like `ybar` except that the bars are horizontal.



```
\usetikzlibrary {patterns}
\tikz \draw[pattern=north west lines] plot[xbar]
   coordinates{(1,0) (0.4,1) (1.7,2) (1.6,3)};
```

`/tikz/ybar interval` (no value)

As `/tikz/ybar`, this options produces vertical bars. However, bars are centered at coordinate *intervals* instead of interval edges, and the bar's width is also determined relatively to the interval's length:



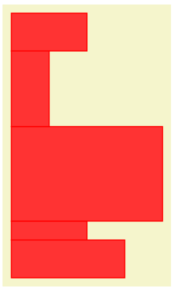
```
\begin{tikzpicture}[ybar interval,x=10pt]
  \draw[color=red,fill=red!80]
    plot coordinates{(0,2) (2,1.2) (3,.3) (5,1.7) (8,.9) (9,.9)};
\end{tikzpicture}
```

Since there are N intervals $[x_i, x_{i+1}]$ for given $N+1$ coordinates, you will always have one coordinate more than bars. The last y value will be ignored.

You can configure relative shifts and relative bar widths, which is explained in the `plothandlers` library documentation, section 65.4. Please refer to page 754.

`/tikz/xbar interval` (no value)

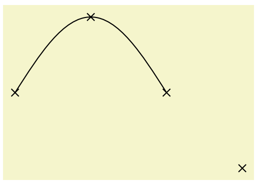
Works like `ybar interval`, but for horizontal bar plots.



```
\begin{tikzpicture}[xbar interval,x=0.5cm,y=0.5cm]
  \draw[color=red,fill=red!80]
    plot coordinates {(3,0) (2,1) (4,1.5) (1,4) (2,6) (2,7)};
\end{tikzpicture}
```

`/tikz/only marks` (no value)

This option causes only marks to be shown; no path segments are added to the actual path. This can be useful for quickly adding some marks to a path.



```
\tikz \draw (0,0) sin (1,1) cos (2,0)
  plot[only marks,mark=x] coordinates{(0,0) (1,1) (2,0) (3,-1)};
```

CHAPTER 4

zTikZ Implement

4.1 Set up Env

4.1.1 Set up (merged)

```
\RequirePackage{l3sys-shell}
\NewDocumentCommand\ztikzMkdir{m}{ \sys_shell_mkdir:n {#1} }
\ExplSyntaxOff
\begingroup\newif\ifztikz
\IfFileExists{./ztikz_output/scripts/sympy_script.py}{}{\ztikztrue}
\ifztikz
% create folders and hash file
\ztikzMkdir{ztikz_output/}
\ztikzMkdir{ztikz_output/gnuplot_data/}
\ztikzMkdir{ztikz_output/python_data/}
\ztikzMkdir{ztikz_output/scripts/}
\ztikzMkdir{ztikz_output/tikz_data/}
\ztikzMkdir{ztikz_output/mma_data/}
\immediate\write18 {touch ztikz_output/ztikz.hash}
```

4.1.2 load module

```
\cs_new_nopar:Npn \g__ztikz_load_module:n #1 {
  \clist_map_inline:nn {#1} {
    \file_if_exist_input:nF {modules/ztikzmodule.##1.tex} {}
  }
}
\NewDocumentCommand\ztikzLoadModule{m}{
  \g__ztikz_load_module:n {#1}\ExplSyntaxOff
}
```

4.1.3 config (merged)

```
% data or picture path
\tl_const:Nn \g__ztikz_gnu_path_tl      {./ztikz_output/gnuplot_data}
\tl_const:Nn \g__ztikz_python_path_tl   {./ztikz_output/python_data}
\tl_const:Nn \g__ztikz_scripts_path_tl  {./ztikz_output/scripts}
\tl_const:Nn \g__ztikz_tikz_path_tl     {./ztikz_output/tikz_data}
\tl_const:Nn \g__ztikz_mma_path_tl      {./ztikz_output/mma_data}

% index of output data by gnuplot
\int_new:N   \g__gnu_data_index_int
\int_new:N   \g__sympy_index_int
\int_new:N   \g__picture_index_int
\int_new:N   \g__mma_index_int
\int_new:N   \g__tikz_env_index_int
\int_gadd:Nn \g__mma_index_int {1}

% bool to control \PlotPrecise cmd
\bool_new:N \g__plot_precise_bool
\bool_new:N \g__contour_precise_bool
\bool_new:N \g__param_precise_bool
\bool_new:N \g__polar_precise_bool
\bool_new:N \g__hash_change_bool
\bool_gset_true:N \g__hash_change_bool

% io read/write interface
\ior_new:N \g__file_read_ior
\tl_new:N  \g__file_content_tl
```

4.2 scripts

4.2.1 python scripts

```
% writing source code
\begin{filecontents}[noheader]{./ztikz_output/scripts/python_script.py}
from numpy import *
from scipy import *



# ==> echo to file
Float_res = 2**10

```

```
# ==> write output
filename = "./ztikz_output/python_data/PyFloat.out"
with open(filename, 'w') as file:
    file.write(str(Float_res)+"\n")
\end{filecontents}

\begin{filecontents}[noheader]{./ztikz_output/scripts/sympy_script.py}
from sympy import *


# ==> symbols declearation
x, y, z, u, v, t = symbols('x y z u v t')

# ==> echo to file
F_res = integrate(sin(x)/x, (x, -oo, oo))

# ==> write output
filename = './ztikz_output/python_data/sympy.out'
with open(filename, 'w') as file:
    file.write(latex(F_res)+"\n")
\end{filecontents}
```

4.2.2 gnuplot scripts

```
\begin{filecontents}[noheader]{./ztikz_output/scripts/plot.gp}
set table "./ztikz_output/gnuplot_data/gnu_data.table"
set format "%.5f"
set samples 100


# ==> specific a 2d function
f(x) = x
set xr [-6:-1]
plot f(x)
\end{filecontents}

\begin{filecontents}[noheader]{./ztikz_output/scripts/contour_plot.gp}
set table "./ztikz_output/gnuplot_data/gnu_data.table"
set samples 100,100
set isosamples 100,100
set format "%.5f"
set cont base
set cntrparam levels discrete 0,0
```

```
unset surface


# ==> contour plot a function
set xr [-4:4]
set yr [*:*]
set zr [*:*]
f(x, y) = y-3*sin(1/x)
splot f(x,y)
\end{filecontents}

\begin{filecontents}[noheader]{./ztikz_output/scripts/param_plot.gp}
set table "./ztikz_output/gnuplot_data/gnu_data.table"
set format "%.5f"
set samples 100
set parametric


# ==> specific a parametric function
set trange [0:2*pi]
plot sin(t), cos(t)
\end{filecontents}

\begin{filecontents}[noheader]{./ztikz_output/scripts/polar_plot.gp}
set table "./ztikz_output/gnuplot_data/gnu_data.table"
set format "%.5f"
set samples 100
set polar


# ==> polar plot, default varible name 't'
set trange [0:12*pi]
plot t
\end{filecontents}
\fi\endgroup
\ExplSyntaxOn
```

4.3 tikz

4.3.1 Cache mechanism

```
% generate md5 hash (history) of a file
\ior_new:N \g__ztikz_file_ior
\seq_new:N \g__ztikz_file_hash_seq
\cs_new_protected:Nn \ztikz_file_read_lines_cs:n {
    \ior_open:Nn \g__ztikz_file_ior {#1}
    \seq_gclear:N \g__ztikz_file_hash_seq
    \ior_str_map_inline:Nn \g__ztikz_file_ior
      {
        \seq_gput_right:Nx \g__ztikz_file_hash_seq
          { \tl_trim_spaces:n {##1} }
      }
    \ior_close:N \g__ztikz_file_ior
}
\cs_generate_variant:Nn \ztikz_file_read_lines_cs:n { e }

% check if code changed (compare hash value of source code)
\seq_new:N \l__ztikz_hash_seq
\cs_new_protected:Npn \ztikz_hash_if_change_cs:n #1 {
    % Param: #1 -> file name of source code
    % read from hash list and remove duplicate item
    \ztikz_file_read_lines_cs:e { ./ztikz_output/ztikz.hash }
    \seq_gremove_duplicates:N \g__ztikz_file_hash_seq
    % check if in
    \file_get_mdfive_hash:nN {#1} \l__ztikz_hash_seq
    \typeout{current~hash~is~\l__ztikz_hash_seq}
    \seq_if_in:NVTF \g__ztikz_file_hash_seq \l__ztikz_hash_seq {
        \bool_gset_false:N \g__hash_change_bool
    }{
        \bool_gset_true:N \g__hash_change_bool
        \typeout{current~hash~is~unique~-->~recorded}
        \sys_shell_now:x { echo~ \l__ztikz_hash_seq >> ./ztikz_output/ztikz.hash}
    }
}
\cs_generate_variant:Nn \ztikz_hash_if_change_cs:n { x }
```

4.3.2 Tikz externalize

```latex
% set options
\cs_new_protected:Npn \ztikz_define_option:n
    { \keys_define:nn { ztikz / option } }

\ztikz_define_option:n {
    external    .bool_set:N = \l__ztikz_external_bool,
    external    .initial:n  = { true },
}
\ProcessKeysOptions {ztikz / option}

\bool_if:NT \l__ztikz_external_bool {
    \usetikzlibrary{external}
    \tikzexternalize[prefix=ztikz_output/tikz_data/]
}
\end{codeprint}

\subsection{ShowPoint}
\begin{minted}[linenos=true]{latex}
% ==> show point
\keys_define:nn { ztikz / point }{
    type    .str_set:N  = \l__point_type_str,
    type    .initial:n = { * },
    radius  .dim_set:N = \l__point_radius_dim,
    radius  .initial:n = { 1pt },
    color   .tl_set:N  = \l__point_color_tl,
    color   .initial:n = { black },
    opacity .tl_set:N  = \l__point_opacity_tl,
    opacity .initial:n = { 1 },
    rotate  .fp_set:N  = \l__point_rotate_angle,
    rotate  .initial:n = { 0 },
}
\NewDocumentCommand\ShowPoint{ O{}mO{}O{} }{
    \group_begin:
    \keys_set:nn { ztikz / point } { #1 }
    \seq_set_split:Nnn \l__point_list_seq { ; }{#2}
    \seq_set_split:Nnn \l__point_label_seq { ; }{#3}
    \int_step_inline:nnnn {1}{1}{\seq_count:N \l__point_list_seq}{
        \draw plot [
            only~ marks,
            mark = \str_use:N \l__point_type_str,
            mark~ size = \dim_use:N \l__point_radius_dim,
            mark~ options = {
                rotate  = \fp_use:N \l__point_rotate_angle,
                opacity = \tl_use:N \l__point_opacity_tl,
                color   = \tl_use:N \l__point_color_tl,
                ball~ color = \tl_use:N \l__point_color_tl,
            }
        ] coordinates{\seq_item:Nn \l__point_list_seq{##1}} node[#4]{\seq_item:Nn
        \l__point_label_seq{##1}};
    }
    \group_end:
}

% ==> show grid
\NewDocumentCommand\ShowGrid{ O{color=gray, very~ thin, step=1}m }{
    \seq_set_split:Nnn \l__grid_param_ii_seq { ; }{#2}
    \draw[#1] \seq_item:Nn \l__grid_param_ii_seq{1}  grid \seq_item:Nn
    \l__grid_param_ii_seq{2};
}
```

4.3.3 ShowAxis

```
% ==> show axis
\keys_define:nn { ztikz / axis }{
    % basic tick args
    tickStart       .fp_set:N   = \l__start_fp,
    tickStart       .initial:n  = { -5 },
    tickEnd         .fp_set:N   = \l__end_fp,
    tickEnd         .initial:n  = { 5 },
    axisRotate      .fp_set:N   = \l__axis_rotate_angle,
    axisRotate      .initial:n  = { 0 },
    % tick dimension spec
    mainStep        .fp_set:N   = \l__main_step_fp,
    mainStep        .initial:n  = { 1.0 },
    subStep         .fp_set:N   = \l__sub_step_fp,
    subStep         .initial:n  = { 0.1 },
    mainTickLabel   .tl_set:N   = \l__main_tick_label_tl,
    mainTickLabel   .initial:n  = { \fp_use:N {\CurrentFp} },
    tickLabelShift  .dim_set:N  = \l__tick_label_shift_dim,
    tickLabelShift  .initial:n  = { 0pt },
    mainTickLength  .dim_set:N  = \l__main_tick_length_dim,
    mainTickLength  .initial:n  = { 4pt },
    subTickLength   .dim_set:N  = \l__sub_tick_length_dim,
    subTickLength   .initial:n  = { 2pt },
    mainTickLabelPosition .tl_set:N  = \l__main_tick_label_position_tl,
    mainTickLabelPosition .initial:n = { below },
    % color spec
    axisColor       .tl_set:N   = \l__axis_color_tl,
    axisColor       .initial:n  = { black },
    mainTickColor   .tl_set:N   = \l__main_tick_color_tl,
    mainTickColor   .initial:n  = { black },
    subTickColor    .tl_set:N   = \l__sub_tick_color_tl,
    subTickColor    .initial:n  = { black },
    mainTickLabelColor .tl_set:N  = \l__main_tick_label_color_tl,
    mainTickLabelColor .initial:n = { black },
    % tick cross type spec
    tickStyle       .choice:,
    tickStyle/cross .code:n     = \tl_set:Nn \l__tick_spec_tl { cross },
    tickStyle/above .code:n     = \tl_set:Nn \l__tick_spec_tl { above },
    tickStyle/below .code:n     = \tl_set:Nn \l__tick_spec_tl { below },
}
% ticks style
\tl_new:N  \l__tick_type_tl    % `main' or `sub'
\tl_new:N  \l__tick_spec_tl    % `cross', `above' or `below
\tl_new:N  \l__tick_color_tl
\dim_new:N \l__tick_length_dim
\tl_new:N  \l__node_text_tl
% draw ticks (main or sub)
\cs_new_protected:Npn \ztikz_draw_axis_ticks_cs:n #1 {
    \str_case:NnT \l__tick_type_tl {
        {main}{
            \dim_set_eq:NN \l__tick_length_dim\l__main_tick_length_dim
            \tl_set:NV \l__tick_color_tl\l__main_tick_color_tl
            \tl_set:Nn \l__node_text_tl {\tl_use:N \l__main_tick_label_tl}
        }
        {sub}{
            \dim_set_eq:NN \l__tick_length_dim\l__sub_tick_length_dim
            \tl_set:NV \l__tick_color_tl \l__sub_tick_color_tl
            \tl_set:Nn \l__node_text_tl {}
        }
    }{}
    \str_case:VnT \l__tick_spec_tl {
        {cross}{
            \draw[\tl_use:N \l__tick_color_tl]
                (#1, 0)++(0, \dim_eval:n {\l__tick_length_dim/2}) -- ++(0, \dim_eval:n
                {-\l__tick_length_dim})
                node[\tl_use:N \l__main_tick_label_position_tl]
                {\textcolor{\tl_use:N \l__main_tick_label_color_tl}{\tl_use:N
                \l__node_text_tl}};
        }
        {above}{
            \draw[\tl_use:N \l__tick_color_tl] (#1, 0) -- ++(0, \dim_eval:n {\l__tick_length_dim/2})
                node[\tl_use:N \l__main_tick_label_position_tl]
                {\textcolor{\tl_use:N \l__main_tick_label_color_tl}{\tl_use:N \l__node_text_tl}};
        }
        {below}{
            \draw[\tl_use:N \l__tick_color_tl] (#1, 0) -- ++(0, \dim_eval:n {-\l__tick_length_dim/2})
                node[\tl_use:N \l__main_tick_label_position_tl=\dim_use:N \l__tick_label_shift_dim]
                {\textcolor{\tl_use:N \l__main_tick_label_color_tl}{\tl_use:N \l__node_text_tl}};
        }
    }{}
}
% draw axis
\int_new:N \l__substep_num_int
\fp_new:N \CurrentFp
\NewDocumentCommand\ShowAxis{O{}m}{
        \group_begin:
        \keys_set:nn { ztikz / axis } { #1 }
        \seq_set_split:Nnn \l__points_seq { ; }{#2}
        \begin{scope}[rotate=\fp_use:N \l__axis_rotate_angle]
        \draw[->, \tl_use:N \l__axis_color_tl] \seq_item:Nn \l__points_seq{1} -- \seq_item:Nn \l__points_seq{2};
        % draw ticks
        \fp_step_inline:nnnn {\fp_eval:n {\l__start_fp}}{\fp_use:N \l__main_step_fp}{\fp_use:N \l__end_fp}{
            % main ticks
            \tl_set:Nn \l__tick_type_tl {main}
            \fp_gset:Nn \CurrentFp {##1}
            \ztikz_draw_axis_ticks_cs:n {##1}
            % sub ticks
            \tl_set:Nn \l__tick_type_tl {sub}
            \int_set:Nn \l__substep_num_int {\fp_eval:n {floor(\l__main_step_fp/\l__sub_step_fp)}}
            \fp_compare:nNnTF {##1}<{\fp_eval:n {floor(\l__end_fp)}}{
                \fp_step_function:nnnN
                    {\fp_eval:n {##1+\l__sub_step_fp}}
                    {\fp_use:N \l__sub_step_fp}
                    {\fp_eval:n {##1+\l__substep_num_int*\l__sub_step_fp}}
                    \ztikz_draw_axis_ticks_cs:n
            }{}
        }
        \end{scope}
        \group_end:
}
\NewDocumentCommand{\xAxis}{O{-2}O{8}}{
    \ShowAxis[
        tickStart=\fp_eval:n {#1+1}, tickEnd=\fp_eval:n {#2-0.75},
        mainStep=1, subStep=.25,
        axisRotate=0, axisColor=black,
        mainTickColor=black, subTickColor=black,
        mainTickLenght=10pt, subTickLenght=5pt,
        tickLabelShift=0pt, tickStyle=below,
        mainTickLabelPosition=below
    ]{(#1, 0); (#2, 0)}
}
\NewDocumentCommand{\yAxis}{O{-2}O{8}}{
    \ShowAxis[
        tickStart=\fp_eval:n {#1+1}, tickEnd=\fp_eval:n {#2-0.75},
        mainStep=1, subStep=.25,
        axisRotate=90, axisColor=black,
        mainTickColor=black, subTickColor=black,
        mainTickLenght=10pt, subTickLenght=5pt,
        tickLabelShift=0pt, tickStyle=above,
        mainTickLabelPosition=left
    ]{(#1, 0); (#2, 0)}
}

% ==> show intersection
\NewDocumentCommand\ShowIntersection{ omm }{
    \seq_set_split:Nnn \l__intersection_num_seq { ; }{#2}
    % get all intersections
    \path[name~ intersections={of=\seq_item:Nn \l__intersection_num_seq{1}~ and~
    \seq_item:Nn \l__intersection_num_seq{2}}];
    % show all intersections by \ShowPoint
    \int_step_inline:nnnn {1}{1}{#3}{
        \ShowPoint[#1]{(intersection-##1)}
    }
}
```

4.3.4 Polygon

```
% ==> polygon
\keys_define:nn { ztikz / polygon }{
    radius       .fp_set:N  = \l__polygon_radius_fp,
    radius       .initial:n = { 1 },
    edgeColor    .tl_set:N  = \l__polygon_edge_color_tl,
    edgeColor    .initial:n = { black },
    fillColor    .tl_set:N  = \l__polygon_fill_color_tl,
    fillColor    .initial:n = { },
    fillOpacity  .fp_set:N  = \l__polygon_fill_opacity_fp,
    fillOpacity  .initial:n = { 0 },
    rotate       .fp_set:N  = \l__polygon_rotate_angle,
    rotate       .initial:n = { 0 },
    shift        .tl_set:N  = \l__polygon_shift_tl,
    shift        .initial:n = { (0,0) },
    marker       .tl_set:N  = \l__polygon_marker_option_tl,
    marker       .initial:n = { },
}
\tl_new:N \l__poly_path_tl
\NewDocumentCommand\Polygon{ O{}m }{
    \group_begin:
    \keys_set:nn { ztikz / polygon } { #1 }
    % strip '(' and ')'
    \tl_replace_once:Nnn \l__polygon_shift_tl{(}{}
    \tl_replace_once:Nnn \l__polygon_shift_tl{)}{}
    \coordinate (mv) at (\tl_use:N \l__polygon_shift_tl);
    % create polygon
    \begin{scope}[shift=(mv), rotate=\fp_use:N \l__polygon_rotate_angle]
    \int_step_inline:nnn {1}{#2}{
        % draw edges
        \fp_set:Nn \l_angle_fp {360/#2*##1*\c_one_degree_fp}
        \fp_set:Nn \l_angle_next_fp {360/#2*(##1+1)*\c_one_degree_fp}
        \draw[\tl_use:N \l__polygon_edge_color_tl] (\fp_eval:n
        {\l__polygon_radius_fp*cos(\l_angle_fp)},      \fp_eval:n
        {\l__polygon_radius_fp*sin(\l_angle_fp)})
            --(\fp_eval:n {\l__polygon_radius_fp*cos(\l_angle_next_fp)}, \fp_eval:n
            {\l__polygon_radius_fp*sin(\l_angle_next_fp)});
        % fill polygon path
        \int_compare:nNnTF {##1}<{#2}{
            \tl_put_right:Nn \l__poly_path_tl {(p##1)--}
        }{
            \tl_put_right:Nn \l__poly_path_tl {(p##1)--cycle}
        }
        % mark coordinates
        \coordinate (p##1) at (\fp_eval:n {\l__polygon_radius_fp*cos(\l_angle_fp)},
        \fp_eval:n {\l__polygon_radius_fp*sin(\l_angle_fp)});
    }
    % fill polygon (none-color -> opacity=1; or opacity=.75)
    \tl_if_empty:NTF \l__polygon_fill_color_tl {
        \fp_set:Nn \l__polygon_fill_opacity_fp {0}
    }{
        \fp_set:Nn \l__polygon_fill_opacity_fp {.75}
    }
    \fill [\tl_use:N \l__polygon_fill_color_tl, fill~opacity=\fp_use:N
    \l__polygon_fill_opacity_fp] \l__poly_path_tl;
    % show markers
    \int_step_inline:nnn {1}{#2}{
        \ShowPoint[\l__polygon_marker_option_tl]{(p##1)}
    }
    \end{scope}
    \group_end:
}
```

4.3.5 Plot precise

```
% change precise of each plot type
\NewDocumentCommand\PlotPrecise{ mO{once}m }{
    \str_if_eq:nnTF {#2}{once}{
        \bool_gset_true:c {g__#1_precise_bool}
    }{\relax}
    % check plot type
    \str_case:nnF {#1}{
        {plot}{
            \sys_shell_now:x {sed~ -i~ "3s|set~ samples~ .*|set~ samples~ #3|"~
            \g__ztikz_scripts_path_tl/plot.gp       }
        }
        {param}{
            \sys_shell_now:x {sed~ -i~ "3s|set~ samples~ .*|set~ samples~ #3|"~
            \g__ztikz_scripts_path_tl/param_plot.gp }
        }
        {polar}{
            \sys_shell_now:x {sed~ -i~ "3s|set~ samples~ .*|set~ samples~ #3|"~
            \g__ztikz_scripts_path_tl/param_plot.gp }
        }
        {contour}{
            \sys_shell_now:x {sed~ -i~ "2s|set~ samples~ .*|set~ samples~ #3,#3|"~
            \g__ztikz_scripts_path_tl/contour_plot.gp}
            \sys_shell_now:x {sed~ -i~ "3s|set~ isosamples~ .*|set~ isosamples~ #3,#3|"~
            \g__ztikz_scripts_path_tl/contour_plot.gp}
        }
    }{
        \msg_new:nnn {ztikz}{ztikz-plot-type}{plot~type~support~is:~'plot',~'param',~'po
        lar',~'contour'}
        \msg_error:nn {ztikz}{ztikz-plot-type}
    }
}
```

4.3.6 Curve plot

```
% escape shell commands
\cs_generate_variant:Nn \sys_shell_mv:nn {xx}
\cs_generate_variant:Nn \sys_shell_now:n {x}

% gnu data reference
\NewDocumentCommand\gnudata{m}{
    \tl_use:N \g__ztikz_gnu_path_tl/gnu_data_ \int_use:N \g__tikz_env_index_int
    _#1.table
}

% gnuplot data rename, plot and precise reset
\cs_new_protected:Npn \ztikz_gnu_data_plot_cs:nnn #1#2#3 {
    % rename data file
    \int_gadd:Nn \g__gnu_data_index_int {1}
    \tl_set:Nx \l__gnu_data_new_name_tl {
        gnu_data_\int_use:N \g__tikz_env_index_int _
        \int_use:N \g__gnu_data_index_int.table
    }
    \tl_set:Nx \l__gnu_data_full_path_tl
    {\g__ztikz_gnu_path_tl/\l__gnu_data_new_name_tl}
    \sys_shell_mv:xx {\g__ztikz_gnu_path_tl/gnu_data.table}
                    {\l__gnu_data_full_path_tl}
    % plot data file
    \tl_if_empty:nTF {#3}{
        \draw[#2] plot[smooth] file {\l__gnu_data_full_path_tl};
    }{
        \group_begin:
        \keys_set:nn { ztikz / point } { #3 }
        \draw plot [
            mark = \str_use:N \l__point_type_str,
            mark~ size = \dim_use:N \l__point_radius_dim,
            mark~ options = {
                rotate  = \fp_use:N \l__point_rotate_angle,
                opacity = \tl_use:N \l__point_opacity_tl,
                color   = \tl_use:N \l__point_color_tl,
                ball~ color = \tl_use:N \l__point_color_tl,
            }
        ] file {\l__gnu_data_full_path_tl};
        \group_end:
    }
    % reset precise (default 300 for plot precise)
    \bool_if:cTF {g__#1_precise_bool}{
        \PlotPrecise{#1}{300}
    }{\relax}
}

% ==> simple 2d function
\NewDocumentCommand\Plot{ O{-5:5}O{color=black}O{}m }{
    % sed gnuplot scripts
    \sys_shell_now:x {sed~ -i~ "8s|set~ xr~ .*|set~ xr~ [#1]|"~
    \g__ztikz_scripts_path_tl/plot.gp}
    \sys_shell_now:x {sed~ -i~ "7s|f(x)~ =~ .*|f(x)~ =~ #4|"  ~
    \g__ztikz_scripts_path_tl/plot.gp}
    \sys_shell_now:x {gnuplot~
    \g__ztikz_scripts_path_tl/plot.gp}
    % plot data, reset precise
    \ztikz_gnu_data_plot_cs:nnn {plot}{#2}{#3}
}

% ==> implicit 2d function
\NewDocumentCommand\ContourPlot{ O{-5:5; -5:5}O{color=black}O{}m }{
^^I% split the first param
    \seq_set_split:Nnn \l__plot_domain_seq { ; }{#1}
    \tl_set:Nn \l__y_domain_tl {\seq_item:Nn \l__plot_domain_seq{2}}
    \exp_args:Nx \tl_if_blank:nTF {\seq_item:Nn \l__plot_domain_seq{2}}{
        \tl_set:Nn \l__y_domain_tl {*:*}
    }{
        \tl_set:Nn \l__y_domain_tl {\seq_item:Nn \l__plot_domain_seq{2}}
    }
    % sed gnuplot scripts
    \sys_shell_now:x {sed~ -i~ "11s|set~ xr~ .*|set~ xr~ [\seq_item:Nn
    \l__plot_domain_seq{1}]|"~ \g__ztikz_scripts_path_tl/contour_plot.gp}
    \sys_shell_now:x {sed~ -i~ "12s|set~ yr~ .*|set~ yr~ [\tl_use:N \l__y_domain_tl]|"~
    \g__ztikz_scripts_path_tl/contour_plot.gp}
    \sys_shell_now:x {sed~ -i~ "14s|f(x,~ y)~ =~ .*|f(x,~ y)~ =~ #4|"~
    \g__ztikz_scripts_path_tl/contour_plot.gp}
    \sys_shell_now:x {gnuplot~
    \g__ztikz_scripts_path_tl/contour_plot.gp}
    % plot data, reset precise
    \ztikz_gnu_data_plot_cs:nnn {contour}{#2}{#3}
}

```

```
% ==> parametric 2d function
\NewDocumentCommand\ParamPlot{ O{0:2*pi}O{color=black}O{}m }{
    % sed gnuplot scripts
    \sys_shell_now:x {sed~ -i~ "8s|set~ trange~ .*|set~ trange~ [#1]|"~
    \g__ztikz_scripts_path_tl/param_plot.gp}
    \sys_shell_now:x {sed~ -i~ "9s|plot~ .*|plot~ #4|"~
    \g__ztikz_scripts_path_tl/param_plot.gp}
    \sys_shell_now:x {gnuplot~
    \g__ztikz_scripts_path_tl/param_plot.gp}
    % plot data, reset precise
    \ztikz_gnu_data_plot_cs:nnn {param}{#2}{#3}
}

% ==> polar 2d function
\NewDocumentCommand\PolarPlot{ O{0:2*pi}O{color=black}O{}m }{
    % sed gnuplot scripts
    \sys_shell_now:x {sed~ -i~ "8s|set~ trange~ .*|set~ trange~ [#1]|"~
    \g__ztikz_scripts_path_tl/polar_plot.gp}
    \sys_shell_now:x {sed~ -i~ "9s|plot~ .*|plot~ #4|"~
    \g__ztikz_scripts_path_tl/polar_plot.gp}
    \sys_shell_now:x {gnuplot~
    \g__ztikz_scripts_path_tl/polar_plot.gp}
    % plot data, reset precise
    \ztikz_gnu_data_plot_cs:nnn {polar}{#2}{#3}
}

% ==> plot 3d
\keys_define:nn { ztikz/gnuplot/plotz }{
  domain      .tl_set:N   = \l__ztikz_gnuplot_plotz_domain_tl,
  domain      .initial:n  = {-5:5; -5:5},
  palette     .tl_set:N   = \l__ztikz_gnuplot_plotz_palette_tl,
  palette     .initial:n  = {rgbformulae~ 22,13,-31},
  width       .dim_set:N  = \l__ztikz_gnuplot_plotz_width_dim,
  width       .initial:n  = {0.75\linewidth},
}
\NewDocumentCommand\Plotz{ O{}m }{
  \group_begin:
  \keys_set:nn { ztikz/gnuplot/plotz } { #1 }
^^I% split the first param
  \seq_set_split:Nnx \l__plot_domain_seq { ; }{\l__ztikz_gnuplot_plotz_domain_tl}
  \tl_set:Nn \l__y_domain_tl {\seq_item:Nn \l__plot_domain_seq{2}}
  \exp_args:Nx \tl_if_blank:nTF {\seq_item:Nn \l__plot_domain_seq{2}}{
    \tl_set:Nn \l__y_domain_tl {*:*}
  }{
    \tl_set:Nn \l__y_domain_tl {\seq_item:Nn \l__plot_domain_seq{2}}
  }
  % sed gnuplot scripts
  \sys_shell_now:x {sed~ -i~ "18s|set~ palette~ .*|set~ palette~ \tl_use:N
  \l__ztikz_gnuplot_plotz_palette_tl|"~  \g__ztikz_scripts_path_tl/plot_3d.gp}
  \sys_shell_now:x {sed~ -i~ "23s|set~ xr~ .*|set~ xr~ [\seq_item:Nn
  \l__plot_domain_seq{1}]|"~ \g__ztikz_scripts_path_tl/plot_3d.gp}
  \sys_shell_now:x {sed~ -i~ "24s|set~ yr~ .*|set~ yr~ [\tl_use:N \l__y_domain_tl]|"~
  \g__ztikz_scripts_path_tl/plot_3d.gp}
  \sys_shell_now:x {sed~ -i~ "25s|splot~ .*|splot~ #2|"~
  \g__ztikz_scripts_path_tl/plot_3d.gp}
  \sys_shell_now:x {gnuplot~
  \g__ztikz_scripts_path_tl/plot_3d.gp}
  % include generate pdf
  \tl_set:Nn \l_tmpa_tl {./ztikz_output/gnuplot_data/plot_3d_\int_use:N
  \g__gnu_plotz_index_int.pdf}
  \sys_shell_mv:nx {./ztikz_output/gnuplot_data/plot_3d.pdf}{\l_tmpa_tl}
  \includegraphics[width=\dim_use:N \l__ztikz_gnuplot_plotz_width_dim]{\l_tmpa_tl}
  \int_gadd:Nn \g__gnu_plotz_index_int {1}
  \group_end:
}
```

4.3.7 Statistic Plot

```
% ==> statistic plot function
\cs_new_protected:Npn \ztikz_statistic_plot_cs:nnnn #1#2#3#4 {
    \tl_if_empty:nTF {#3}{
        \draw[#2] plot[#1] file {#4};
    }{
        \group_begin:
        \keys_set:nn { ztikz / point } { #3 }
        \draw[#2] plot [
            % stairs options
            #1,
            % marker options
            mark = \str_use:N \l__point_type_str,
            mark~ size = \dim_use:N \l__point_radius_dim,
            mark~ options = {
                rotate  = \fp_use:N \l__point_rotate_angle,
                opacity = \tl_use:N \l__point_opacity_tl,
                color   = \tl_use:N \l__point_color_tl,
                ball~ color = \tl_use:N \l__point_color_tl,
            }
        ] file {#4};
        \group_end:
    }
}
\cs_generate_variant:Nn \ztikz_statistic_plot_cs:nnnn {xnnn}

\tl_new:N \l__tempa_tl
\tl_new:N \l__tempb_tl
\seq_new:N \l__statistic_option_tl
% 1. \ListPlot --> set opcity=0 in the above Plot commands
% 2. stairs plot
\NewDocumentCommand\StairsPlot{ O{plot-left;jump-left}O{color=black}O{}m }{
    \seq_set_split:Nnn \l__statistic_option_tl { ; }{#1}
    \str_case:enF {\seq_item:Nn \l__statistic_option_tl{1}}{
        {plot-left}{\tl_set:Nn \l__tempa_tl {const~plot~mark~left}}
        {plot-right}{\tl_set:Nn \l__tempa_tl {const~plot~mark~right}}
        {plot-mid}{\tl_set:Nn \l__tempa_tl {const~plot~mark~mid}}
        {}{\tl_set:Nn \l__tempa_tl {}}
    }{
        \msg_new:nnn {ztikz}{ztikz-stairs-plot}{current~stairs~plot~type~is:~'#1'~,~
        invalide}
        \msg_error:nn {ztikz}{ztikz-stairs-plot}
    }
    \str_case:enF {\seq_item:Nn \l__statistic_option_tl{2}}{
        {jump-left}{\tl_set:Nn \l__tempb_tl {jump~mark~left}}
        {jump-right}{\tl_set:Nn \l__tempb_tl {jump~mark~right}}
        {jump-mid}{\tl_set:Nn \l__tempb_tl {jump~mark~mid}}
        {}{\tl_set:Nn \l__tempb_tl {}}
    }{
        \msg_new:nnn {ztikz}{ztikz-stairs-plot}{current~stairs~jump~type~is:~'#1'~,~
        invalide}
        \msg_error:nn {ztikz}{ztikz-stairs-plot}
    }
    \ztikz_statistic_plot_cs:xnnn {\l__tempa_tl,\l__tempb_tl}{#2}{#3}{#4}
}
% 3. stem plot
\NewDocumentCommand\StemPlot{ O{x}O{color=black}O{}m }{
    \str_case:enF {#1}{
        {x}{\tl_set:Nn \l__tempa_tl {ycomb}}
        {y}{\tl_set:Nn \l__tempa_tl {xcomb}}
        {o}{\tl_set:Nn \l__tempa_tl {polar~ comb}}
        {}{\tl_set:Nn \l__tempa_tl  {ycomb}}
    }{
        \msg_new:nnn {ztikz}{ztikz-stem-plot}{current~stem~plot~type~is:~'#1'~,~
        invalide}
        \msg_error:nn {ztikz}{ztikz-stem-plot}
    }
    \ztikz_statistic_plot_cs:xnnn {\l__tempa_tl}{#2}{#3}{#4}
}
% 4. bar plot
\NewDocumentCommand\BarPlot{ O{ybar}O{color=black}O{}m }{
    \str_case:enF {#1}{
        {x}{\tl_set:Nn \l__tempa_tl {ybar}}
        {y}{\tl_set:Nn \l__tempa_tl {xbar}}
        {xc}{\tl_set:Nn \l__tempa_tl {ybar~ interval}}
        {yc}{\tl_set:Nn \l__tempa_tl {xbar~ interval}}
        {}{\tl_set:Nn \l__tempa_tl  {ybar}}
    }{
        \msg_new:nnn {ztikz}{ztikz-bar-plot}{current~bar~plot~type~is:~'#1'~,~ invalide}
        \msg_error:nn {ztikz}{ztikz-bar-plot}
    }
    \ztikz_statistic_plot_cs:xnnn {\l__tempa_tl}{#2}{#3}{#4}
}
```

4.3.8 ShadePlot

➠ pgf based (removed)

```
% 5. shade plot
\NewDocumentCommand\ShadePlot{ O{defaultMode}O{2pt}m }{
    \path [shading=#1, shading~ path={draw=transparent!0, line~ width=#2}] plot file
    {#3};
}

% ==> shade curve draw
\ExplSyntaxOff
% ------------------------------------------------------------------------------------
---------
% https://tex.stackexchange.com/questions/497977/tikz-draw-multicolor-curve-with-smooth-
gradient
% ------------------------------------------------------------------------------------
---------
\newif\iftikz@shading@path
\tikzset {
    fading xsep/.store in=\pgfpathfadingxsep,
    fading ysep/.store in=\pgfpathfadingysep,
    fading sep/.style={fading xsep=#1, fading ysep=#1},
    fading sep=0.0cm,
    shading path/.code= {
        % Prevent this stuff happning recursively.
        \iftikz@shading@path
        \else
            \tikz@shading@pathtrue
            % \tikz@addmode installs the `modes' (e.g., fill, draw, shade)
            % to be applied to the path. It isn't usualy for doing more
            % changes to the path's construction.
            \tikz@addmode{
                \pgfgetpath\pgf@currentfadingpath%
                % Get the boudning box of the current path size including the fading sep
                \pgfextract@process\pgf@fadingpath@southwest{\pgfpointadd{\pgfqpoint{\pgf@pathminx}{\pgf@pathminy}}
                    {\pgfpoint{-\pgfpathfadingxsep}{-\pgfpathfadingysep}}}%
                \pgfextract@process\pgf@fadingpath@northeast{\pgfpointadd{\pgfqpoint{\pgf@pathmaxx}{\pgf@pathmaxy}}
                    {\pgfpoint{\pgfpathfadingxsep}{\pgfpathfadingysep}}}
                % Clear the path
                \pgfsetpath\pgfutil@empty%
                % Interrupt the path and picture to create a fading.
                \pgfinterruptpath
                \pgfinterruptpicture
                \begin{tikzfadingfrompicture}[name=.]
                    \path [shade=none, fill=none, #1] \pgfextra {
                        % Set the softpath. Any transformations in #1 will have no effect.
                        % This will *not* update the bounding box...
                        \pgfsetpath\pgf@currentfadingpath
                        % ...so it is done manually.
                        \pgf@fadingpath@southwest
                        \expandafter\pgf@protocolsizes{\the\pgf@x}{\the\pgf@y}%
                        \pgf@fadingpath@northeast%
                        \expandafter\pgf@protocolsizes{\the\pgf@x}{\the\pgf@y}%
                    };
                    % Now get the bounding of the picture.
                    \xdef\pgf@fadingboundingbox@southwest{\noexpand\pgfqpoint{\the\pgf@picminx}{\the\pgf@picminy}}%
                    \xdef\pgf@fadingboundingbox@northeast{\noexpand\pgfqpoint{\the\pgf@picmaxx}{\the\pgf@picmaxy}}%
                \end{tikzfadingfrompicture}
                \endpgfinterruptpicture
                \endpgfinterruptpath
                % Install a rectangle that covers the shaded/faded path picture.
                \pgfpathrectanglecorners{\pgf@fadingboundingbox@southwest}{\pgf@fadingboundingbox@northeast}
                % Make the fading happen.
                \def\tikz@path@fading{.}
                \tikz@mode@fade@pathtrue
                \tikz@fade@adjustfalse%10p
                % Shift the fading to the mid point of the rectangle
                \pgfpointscale{0.5}{\pgfpointadd{\pgf@fadingboundingbox@southwest}{\pgf@
                fadingboundingbox@northeast}}
                \edef\tikz@fade@transform{shift={(\the\pgf@x,\the\pgf@y)}}
            }
        \fi
    }
}
\def\ztikzShadeMode#1#2#3{
    \pgfutil@tempcnta=0\relax
    \pgfutil@for\pgf@tmp:={#3}\do{\advance\pgfutil@tempcnta by1}
    \ifnum\pgfutil@tempcnta=1\relax
        \edef\pgf@spec{color(0)=(#3);color(100)=(#3)}
    \else
        \pgfmathparse{50/(\pgfutil@tempcnta-1)}\let\pgf@step=\pgfmathresult
        \pgfutil@tempcntb=1\relax
        \pgfutil@for\pgf@tmp:={#3}\do{
            \ifnum\pgfutil@tempcntb=1\relax
            \edef\pgf@spec{color(0)=(\pgf@tmp);color(25)=(\pgf@tmp)}
            \else
            \ifnum\pgfutil@tempcntb<\pgfutil@tempcnta\relax
            \pgfmathparse{25+\pgf@step/4+(\pgfutil@tempcntb-1)*\pgf@step}
            \edef\pgf@spec{\pgf@spec;color(\pgfmathresult)=(\pgf@tmp)}
            \else
            \edef\pgf@spec{\pgf@spec;color(75)=(\pgf@tmp);color(100)=(\pgf@tmp)}
            \fi
            \fi
            \advance\pgfutil@tempcntb by1\relax
        }
    \fi
    \csname pgfdeclare#2shading\endcsname{#1}{100}\pgf@spec
}
\ztikzShadeMode{defaultMode}{horizontal}{white,black}
\ExplSyntaxOn
```

4.4 zdraw module

➠ **l3draw based(using module zdraw)**

4.4.1 set up

```latex
\RequirePackage{l3draw}
\tl_new:N \g__ztikz_zdraw_unit_tl
\tl_gset:Nn \g__ztikz_zdraw_unit_tl {em}
\newcommand{\zdrawSetUnit}[1]{
  \tl_gset:Nn \g__ztikz_zdraw_unit_tl {#1}
}
% default=0.4pt
\newcommand{\zdrawSetPathWidth}[1]{
  \dim_set:Nn \l_draw_default_linewidth_dim {#1}
}
```

```latex
% pre-defined:black, white, red, green, blue, cyan, magenta, yellow
% new-defined:orange, teal, purple, pink, aqua, tan, brown, gray
\NewDocumentCommand{\newcolor}{mmm}{
  \color_set:nnn {#1}{#2}{#3}
}
\color_set:nnn {orange}{HTML}{FFA500}
\color_set:nnn {teal}{HTML}{008080}
\color_set:nnn {purple}{HTML}{800080}
\color_set:nnn {pink}{HTML}{FFC0CB}
\color_set:nnn {aqua}{HTML}{00FFFF}
\color_set:nnn {tan}{HTML}{D2B48C}
\color_set:nnn {brown}{HTML}{A52A2A}
\color_set:nnn {gray}{HTML}{808080}
```

4.4.2 utils

```latex
% arg split
\cs_generate_variant:Nn \int_step_inline:nnn {nen}
\cs_generate_variant:Nn \seq_set_split:Nnn {cnn}
\cs_set:Npn \args_split_cs:nn #1#2 {
  \seq_if_exist:cF {l_#2_seq}{
    \seq_new:c {l_#2_seq}
  }
  \seq_set_split:cnn {l_#2_seq}{,}{#1}
  \int_step_inline:nen {1}{\seq_count:c {l_#2_seq}}{
    \tl_set:Nn \l_tmpa_tl {l__arg_#2_\int_to_roman:n{##1}}
    \exp_args:Nfo \tl_if_exist:cF {\tl_use:N \l_tmpa_tl}{
      \tl_new:c {\tl_use:N \l_tmpa_tl}
    }
    \tl_set:ce {\tl_use:N \l_tmpa_tl}{\seq_item:cn {l_#2_seq}{##1}}
  }
}
```

4.4.3 l3draw alias

```
%% draw env
\NewDocumentEnvironment{zdraw}{O{}}{\ExplSyntaxOn\draw_begin:}{\draw_end:\ExplSyntaxOff}
\NewDocumentEnvironment{zscope}{O{}}{\draw_path_scope_begin:}{\draw_path_scope_end:}

%% cmd alias
% point/line
\cs_new_eq:NN \moveto \draw_path_moveto:n
\cs_new_eq:NN \lineto \draw_path_lineto:n
\cs_new_eq:NN \scolor \color_stroke:n
\cs_new_eq:NN \fcolor \color_fill:n

% base vector
\cs_new_eq:NN \xvec   \draw_xvec:n
\cs_new_eq:NN \yvec   \draw_yvec:n
% both follows ref (0, 0)
\cs_new_eq:NN \polar  \draw_point_polar:nn
\cs_new_eq:NN \coor   \draw_point_vec:nn

% shape and text
\cs_new_eq:NN \rec \draw_path_rectangle_corners:nn
\cs_new_eq:NN \cir \draw_path_circle:nn
\cs_new_eq:NN \newtext    \coffin_new:N
\cs_new_eq:NN \sethtext   \hcoffin_set:Nn
\cs_new_eq:NN \setvtext   \vcoffin_set:Nnn
\cs_new_eq:NN \scaletext  \coffin_scale:Nnn
\cs_new_eq:NN \puttext    \draw_coffin_use:Nnnn

% path operation
% scope
\cs_new_eq:NN \bg \draw_path_scope_begin:
\cs_new_eq:NN \eg \draw_path_scope_end:
% segement connect
\cs_new_eq:NN \capbutt   \draw_cap_butt:
\cs_new_eq:NN \caproun   \draw_cap_round:
\cs_new_eq:NN \caprect   \draw_cap_rectangle:
\cs_new_eq:NN \closepath \draw_path_close:
% transformatio
```

```
\cs_new_eq:NN \shift     \draw_transform_shift:n
\cs_new_eq:NN \xscale    \draw_transform_xscale:n
\cs_new_eq:NN \yscale    \draw_transform_yscale:n
\cs_new_eq:NN \trans     \draw_transform_matrix:nnnn
\NewDocumentCommand{\usepath}{O{draw}}{
  \draw_path_use_clear:n {#1}
}
```

4.4.4 plot function

```
% \function_plot:nnn {<function>}{<domain>}{<style>}
% functin = {<function>}
% domain  = {<x-start>, <x-step>, <x-end>, <y-min>, <y-max>}
% style   = {<unit>, <action>, <color-1>, <color-2>, <gradient-axis>}
\cs_generate_variant:Nn \fp_step_inline:nnnn {eeen}
\cs_set:Npn \function_plot:nnn #1#2#3 {
  % => split arg
  \args_split_cs:nn {#2}{domain}
  \args_split_cs:nn {#3}{style}
  % => draw function
  \draw_begin:
    % normal part
    \str_case:VnT \l__arg_style_ii {
      {stroke}{\exp_args:Nx \color_stroke:n {\tl_use:N \l__arg_style_iii}}
      {draw}{\exp_args:Nx \color_stroke:n {\tl_use:N \l__arg_style_iii}}
      {fill}{\exp_args:Nx \color_fill:n {\tl_use:N \l__arg_style_iii}}
      {clip}{\relax}
    }{
      % => start point
      \tl_set:Nn \l_tmpa_tl {#1}
      \tl_replace_all:Nne \l_tmpa_tl {x}{(\tl_use:N \l__arg_domain_i)}
      \draw_path_moveto:n {\l__arg_domain_i\l__arg_style_i, \l_tmpa_tl\l__arg_style_i}
      % loop to draw path
      \fp_step_inline:eeen {\l__arg_domain_i}{\l__arg_domain_ii}{\l__arg_domain_iii}{
          \tl_set:Nn \l_tmpa_tl {#1}
          \tl_replace_all:Nnn \l_tmpa_tl {x}{(##1)}
          \draw_path_lineto:n {##1 \l__arg_style_i, \l_tmpa_tl \l__arg_style_i}
      }
      \draw_path_use_clear:n {\l__arg_style_ii}
    }
    % shade plot part
    \str_if_eq:VnT \l__arg_style_ii {shade}{
      % start and end point for 'y-axis gradient'
      \tl_if_eq:VnT \l__arg_style_v {y}{
        \tl_if_exist:cF {l__start_tl}{
          \tl_new:N \l__start_tl
          \tl_new:N \l__end_tl
        }
        \tl_set:Nx \l__start_tl {\l__arg_domain_iv}
        \tl_set:Nx \l__end_tl {\l__arg_domain_v}
      }
      % loop to plot segements
      \fp_step_inline:eeen {\l__arg_domain_i}{\l__arg_domain_ii}{\fp_eval:n
      {\l__arg_domain_iii-\l__arg_domain_ii}}{
          \tl_set:Nn \l_tmpa_tl {#1}
          \tl_set:Nn \l_tmpb_tl {#1}
          \tl_replace_all:Nnn \l_tmpa_tl {x}{(##1)}
          \tl_replace_all:Nnn \l_tmpb_tl {x}{(##1+\l__arg_domain_ii)}
          \str_case:VnF \l__arg_style_v {
            {x}{\color_gradient:xxx {
                \fp_eval:n
                {(##1-\l__arg_domain_i)*(100/(\l__arg_domain_iii-\l__arg_domain_i))}
              }{\l__arg_style_iii}{\l__arg_style_iv}}
            {y}{\color_gradient:xxx {
                \fp_eval:n {(\l_tmpa_tl-\l__start_tl)*(100/(\l__end_tl-\l__start_tl))}
              }{\l__arg_style_iii}{\l__arg_style_iv}}
          }{\relax}
          \draw_path_moveto:n {##1 \l__arg_style_i, \l_tmpa_tl \l__arg_style_i}
          \draw_path_lineto:n {(##1+\l__arg_domain_ii) \l__arg_style_i, \l_tmpb_tl
          \l__arg_style_i}
          % \draw_cap_rectangle:
          \draw_cap_round:
          \draw_path_use_clear:n {draw}
      }
    }
  \draw_end:
}

% color gradient
\cs_set:Npn \color_gradient:nnn #1#2#3 {
  \fp_compare:nNnTF {#1}>{100}{
    \exp_args:Nx \color_select:n
    {#2!\fp_eval:n{abs(#1)/(\l__arg_domain_v-\l__arg_domain_iv)}!#3}
  }{
    \fp_compare:nNnTF {#1}<{0}{
      \exp_args:Nx \color_select:n
      {#2!\fp_eval:n{abs(#1)/(\l__arg_domain_v-\l__arg_domain_iv)}!#3}
    }{\color_select:n {#2!#1!#3}}
  }
}
\cs_generate_variant:Nn \color_gradient:nnn {xxx}

% plot functions interface
\keys_define:nn { ztikz/zdraw/zplot }{
  action        .tl_set:N   = \l__ztikz_zdraw_zplot_action_tl,
  action        .initial:n  = {draw},
  domain        .tl_set:N   = \l__ztikz_zdraw_zplot_domain_tl,
  domain        .initial:n  = {-5, 0.1, 5},
  range         .tl_set:N   = \l__ztikz_zdraw_zplot_range_tl,
  range         .initial:n  = {-5, 5},
  startColor    .tl_set:N   = \l__ztikz_zdraw_zplot_startColor_tl,
  startColor    .initial:n  = {black},
  endColor      .tl_set:N   = \l__ztikz_zdraw_zplot_endColor_tl,
  endColor      .initial:n  = {white},
  axis          .tl_set:N   = \l__ztikz_zdraw_zplot_axis_tl,
  axis          .initial:n  = {y}
}
\cs_generate_variant:Nn \function_plot:nnn {nxx}
\NewDocumentCommand\zplot{ mO{} }{
  \group_begin:
  \keys_set:nn { ztikz/zdraw/zplot } { #2 }
  \function_plot:nxx
    {#1}
    {\l__ztikz_zdraw_zplot_domain_tl, \l__ztikz_zdraw_zplot_range_tl}
    {
      \g__ztikz_zdraw_unit_tl, \l__ztikz_zdraw_zplot_action_tl,
      \l__ztikz_zdraw_zplot_startColor_tl, \l__ztikz_zdraw_zplot_endColor_tl,
      \l__ztikz_zdraw_zplot_axis_tl
    }
  \group_end:
}
```

4.4.5 gradient rule

```
\cs_generate_variant:Nn \fp_step_inline:nnnn {nnxn}
\cs_set_nopar:Npn \gradient_rule:nnn #1#2#3 {
  \args_split_cs:nn {#1}{dim}
  \args_split_cs:nn {#2}{color}
  \draw_begin:
  \fp_step_inline:nnxn {0}{#3}{\l__arg_dim_i}{
    \draw_scope_begin:
      \exp_args:Nx \color_select:n {\l__arg_color_ii!\fp_eval:n
      {##1*100/\l__arg_dim_i}!\l__arg_color_i}
      \exp_args:Nnx \draw_path_rectangle_corners:nn
        {(##1) \g__ztikz_zdraw_unit_tl, 0 \g__ztikz_zdraw_unit_tl}
        {(##1+#3) \g__ztikz_zdraw_unit_tl, \l__arg_dim_ii \g__ztikz_zdraw_unit_tl}
      \draw_path_use_clear:n {fill, draw}
    \draw_scope_end:
  }
  \draw_end:
}

% gradient rule interface
\keys_define:nn { ztikz/zdraw/zrule }{
  width         .tl_set:N   = \l__ztikz_zdraw_zplot_width_tl,
  width         .initial:n  = {1},
  height        .tl_set:N   = \l__ztikz_zdraw_zplot_height_tl,
  height        .initial:n  = {1},
  startColor    .tl_set:N   = \l__ztikz_zdraw_zrule_startColor_tl,
  startColor    .initial:n  = {black},
  endColor      .tl_set:N   = \l__ztikz_zdraw_zrule_endColor_tl,
  endColor      .initial:n  = {white},
  step          .fp_set:N   = \l__ztikz_zdraw_zrule_step_fp,
  step          .initial:n  = {0.25}
}
\cs_generate_variant:Nn \gradient_rule:nnn {xxx}
\NewDocumentCommand\zrule{O{}O{}}{
  \group_begin:
  \keys_set:nn { ztikz/zdraw/zrule } { #1 }
  \gradient_rule:xxx
    {\l__ztikz_zdraw_zplot_width_tl, \l__ztikz_zdraw_zplot_height_tl}
    {\l__ztikz_zdraw_zrule_startColor_tl, \l__ztikz_zdraw_zrule_endColor_tl}
    {\l__ztikz_zdraw_zrule_step_fp}
  \group_end:
}
```

4.5 Python module

4.5.1 Matplotlib

```
\cs_generate_variant:Nn \xsim_file_write_start:nn {nx}
\cs_generate_variant:Nn \sys_shell_mv:nn {xx}
% ==> python-matplotlib
```

```
\NewDocumentEnvironment{pyfig}{ O{width=.75\linewidth}m }{
    \newcommand{\pyfile}{#2}
    \xsim_file_write_start:nx {\c_true_bool}{\g__ztikz_python_path_tl/#2}
    }{
    \xsim_file_write_stop:
    % step picture index
    \int_gadd:Nn \g__picture_index_int {1}
    % check if hash changed
    \ztikz_hash_if_change_cs:x {\g__ztikz_python_path_tl/\pyfile}
    \bool_if:NTF \g__hash_change_bool {
        % add save figure to source
        \sys_if_platform_windows:TF {
            \exp_args:Nx \sys_shell_now:n {
                echo~ plt.savefig('\g__ztikz_python_path_tl/\pyfile.pdf')~ >>~
                \g__ztikz_python_path_tl/\pyfile
            }
        }{
            \exp_args:Nx \sys_shell_now:n {
                echo~ "plt.savefig('\g__ztikz_python_path_tl/\pyfile.pdf')"~ >>~
                \g__ztikz_python_path_tl/\pyfile
            }
        }
        % excute python source
        \exp_args:Nx \sys_shell_now:n {python~ \g__ztikz_python_path_tl/\pyfile}
        \includegraphics[#1]{\g__ztikz_python_path_tl/\pyfile.pdf}
        \typeout{Writing~ 'pyfig'~environment~source~to~\tl_use:N
        \g__ztikz_python_path_tl/\pyfile}
    }{
        \includegraphics[#1]{\g__ztikz_python_path_tl/\pyfile.pdf}
        \typeout{skip~recompile~by~python,~using~the~cache~picture~\int_use:N
        \g__picture_index_int}
    }
}
```

4.5.2 Pycode

```
% ==> python-code-env
\NewDocumentEnvironment{pycode}{ m }{
    \newcommand{\pyfile}{#1}
    \xsim_file_write_start:nx {\c_true_bool}{\g__ztikz_python_path_tl/#1}
    }{
    \xsim_file_write_stop:
    % step picture index
    \int_gadd:Nn \g__picture_index_int {1}
    % check if hash changed
    \ztikz_hash_if_change_cs:x {\g__ztikz_python_path_tl/\pyfile}
    \bool_if:NTF \g__hash_change_bool {
        % excute python source
        \exp_args:Nx \sys_shell_now:n {python~ \g__ztikz_python_path_tl/\pyfile}
        \input{\g__ztikz_python_path_tl/\pyfile.out}
        \typeout{Writing~ 'pycode'~environment~source~to~\tl_use:N
        \g__ztikz_python_path_tl/\pyfile}
    }{
        \input{\g__ztikz_python_path_tl/\pyfile.out}
        \typeout{skip~recompile~by~python,~using~the~cache~pycode~result~\int_use:N
        \g__picture_index_int}
    }
}
```

4.5.3 Sympy

```
% ==> python-sympy
\NewDocumentCommand\sympy{m}{
    % step sympy result index
    \int_gadd:Nn \g__sympy_index_int {1}
    \tl_set:Nx \l__current_sympy_index_tl {\int_use:N \g__sympy_index_int}
    % sympy source write
    \sys_shell_now:x {sed~ -i~ "8s|F_res~ =~ .*|F_res~ =~  #1|"~
    \g__ztikz_scripts_path_tl/sympy_script.py}
    % check hash
    \ztikz_hash_if_change_cs:x {\g__ztikz_scripts_path_tl/sympy_script.py}
    \bool_if:NTF \g__hash_change_bool {
        \sys_shell_now:x {python~ \g__ztikz_scripts_path_tl/sympy_script.py}
        \sys_shell_mv:xx
            {\g__ztikz_python_path_tl/sympy.out}
            {\g__ztikz_python_path_tl/sympy_\int_use:N \g__sympy_index_int .out}
        \typeout{using~python~sympy~calculating~question~\l__current_sympy_index_tl ...}
        \exp_args:Nx
        \input{\g__ztikz_python_path_tl/sympy_\l__current_sympy_index_tl.out}
    }{
        \exp_args:Nx
        \input{\g__ztikz_python_path_tl/sympy_\l__current_sympy_index_tl.out}
        \typeout{skip~recompile,~using~the~cache~sympy~result~\l__current_sympy_index_tl
        }
    }
}
```

4.5.4 Python eval

```
% read from external file
\cs_new_protected:Npn \zlatex_Readlines_cs:nn #1#2 {
    \ior_open:Nn \g__file_read_ior {#2}
    \str_case:nnF {#1}{
        {hold}{
            \ior_get:NN \g__file_read_ior \g__file_content_tl
        }
        {str}{
            \ior_str_get:NN \g__file_read_ior \g__file_content_tl
        }
    }{}
    \tl_use:N \g__file_content_tl
}
\cs_generate_variant:Nn \zlatex_Readlines_cs:nn {xx}

% ==> 1-line python command
\NewDocumentCommand\py{O{raw}m}{
    % sympy source write
    \sys_shell_now:x {sed~ -i~ "6s|Float_res~ =~ .*|Float_res~ =~ \tl_to_str:n {#2}|"~
    \g__ztikz_scripts_path_tl/python_script.py}
    % calculation
    \typeout{using~python~float~module~calculating...}
    \sys_shell_now:x {python~ \g__ztikz_scripts_path_tl/python_script.py}
    % using \ior_get:
    \zlatex_Readlines_cs:xx {#1}{\g__ztikz_python_path_tl/PyFloat.out}
    % ---> cause bug that can't write ToC to file
    % \iow_close:N \g__file_read_ior
}
```

4.6 Mathematica

4.6.1 SetUp

```
% temp var for manipulate
\tl_new:N \l__temp_arg_tl
\tl_new:N \l__wolfram_temp_res_tl
\ior_new:N \l__temp_io
\seq_new:N \l__wolfram_temp_res_seq

% macro storage the mma-result
```

```
\NewDocumentCommand\wolframResult{O{raw}O{}}{
    \tl_if_eq:nnTF {#1}{raw}{
        \seq_use:Nnnn \l__wolfram_temp_res_seq {#2}{#2}{#2}
    }{
        \if_is_int:xTF {\int_eval:n {#1}}
            {\seq_item:Nn \l__wolfram_temp_res_seq {#1}}
            {}
    }
}

% check if integer
\prg_new_protected_conditional:Npnn \if_is_int:n #1 { T, F, TF }{
    \regex_match:nnTF { ^[\+\-]?[\d]+$ } {#1} % $
      { \prg_return_true: }
      { \prg_return_false: }
}
\cs_generate_variant:Nn \if_is_int:nTF {xTF}
```

4.6.2 Picture

```
% wolfram graphicx
\NewDocumentEnvironment{wolframGraphics}{ O{width=.75\linewidth}m }{
    \newcommand{\mmafile}{#2}
    \xsim_file_write_start:nx {\c_true_bool}{\g__ztikz_mma_path_tl/#2}
    }{
    \xsim_file_write_stop:
    % step picture index
    \int_gadd:Nn \g__picture_index_int {1}
    % check if hash changed
    \ztikz_hash_if_change_cs:x {\g__ztikz_mma_path_tl/\mmafile}
    \bool_if:NTF \g__hash_change_bool {
        % excute mathematica script
        \exp_args:Nx \sys_shell_now:n {wolframscript~ -script~
        \g__ztikz_mma_path_tl/\mmafile}
        \includegraphics[#1]{\g__ztikz_mma_path_tl/\mmafile.pdf}
        \typeout{Writing~ 'mmafig'~environment~source~to~\tl_use:N
        \g__ztikz_mma_path_tl/\mmafile}
    }{
        \includegraphics[#1]{\g__ztikz_mma_path_tl/\mmafile.pdf}
        \typeout{skip~recompile~by~wolframscript,~using~the~cache~picture~\int_use:N
        \g__picture_index_int}
    }
}
```

4.6.3 Symbols Calculation

```
% input result of wolfram
\cs_generate_variant:Nn \ior_open:Nn {Nx}
\cs_new_protected:Nn \ztikz_wolfram_input_result_cs: {
    % write export command
    \iow_now:Nx \g_tmpa_iow {Export["\g__ztikz_mma_path_tl/mma_res_\int_use:N \g__mma_index_int.txt", TeXResult]}
    \iow_close:N \g_tmpa_iow
    % calculate and input
    \ztikz_hash_if_change_cs:x {\g__ztikz_mma_path_tl/mma_calc_\int_use:N \g__mma_index_int.wls}
    \bool_if:NTF \g__hash_change_bool {
        \sys_shell_now:x {wolframscript~ -script~ \g__ztikz_mma_path_tl/mma_calc_\int_use:N \g__mma_index_int.wls}
        % \exp_args:Nx \input{\g__ztikz_mma_path_tl/mma_res_\int_use:N \g__mma_index_int.txt}
        \typeout{using~wolframscript~calculating~question~\int_use:N \g__mma_index_int...}
    }{
        % \exp_args:Nx \input{\g__ztikz_mma_path_tl/mma_res_\int_use:N \g__mma_index_int.txt}
        \typeout{skip~recompile,~using~the~cache~wolframscript~result~\int_use:N \g__mma_index_int}
    }
    % split result to seq and storage result to a macro
    \ior_open:Nx \l__temp_io {\g__ztikz_mma_path_tl/mma_res_\int_use:N \g__mma_index_int.txt}
    \ior_get:NN  \l__temp_io \l__wolfram_temp_res_tl
    \seq_set_split:NnV \l__wolfram_temp_res_seq{,}\l__wolfram_temp_res_tl
    % step counter, add export command
    \int_gadd:Nn \g__mma_index_int {1}
}
```

4.6.4 Wolfram code

```
% wolfram code
\cs_generate_variant:Nn \iow_open:Nn {Nx}
\NewDocumentCommand\wolfram{O{tex}m}{
    % write mathamatica code
    \iow_open:Nx \g_tmpa_iow {\g__ztikz_mma_path_tl/mma_calc_\int_use:N \g__mma_index_int.wls}
    \str_case:nnF {#1}{
        {tex} {
            \iow_now:Nx \g_tmpa_iow { TeXResult = ToString[TeXForm[#2]]; }
        }
        {text} {
            \iow_now:Nx \g_tmpa_iow { TeXResult = ToString[#2]; }
        }
    }{\relax}
    % cache and input result
    \ztikz_wolfram_input_result_cs:
}
```

4.6.5 Equation Solve

```
% equation solve
\NewDocumentCommand\wolframSolve{O{part}mO{}O{}}{
    % prepend ',' to #4 if not empty
    \tl_if_empty:nTF {#4}{
        \tl_set:Nn \l__temp_arg_tl {}
    }{
        \tl_set:Nn \l__temp_arg_tl {,#4}
    }
    % write mathamatica code
    \iow_open:Nx \g_tmpa_iow {\g__ztikz_mma_path_tl/mma_calc_\int_use:N
    \g__mma_index_int.wls}
    \str_case:nnF {#1}{
        {part} {
            \iow_now:Nx \g_tmpa_iow {
                TeXResult = Row[Solve[#2, {#3} \l__temp_arg_tl]//Flatten, ","]/.{Rule ->
                Equal}//TeXForm//ToString;
            }
        }
        {full} {
            \iow_now:Nx \g_tmpa_iow {
                TeXResult = Row[Solve[#2]//Flatten, ","]/.{Rule ->
                Equal}//TeXForm//ToString;
            }
        }
    }{\relax}
    % cache and input result
    \ztikz_wolfram_input_result_cs:
}

% differential equation solve
```

```
\NewDocumentCommand\wolframDSolve{O{part}mO{}O{}}{
    % prepend ',' to #4 if not empty
    \tl_if_empty:nTF {#4}{
        \tl_set:Nn \l__temp_arg_tl {}
    }{
        \tl_set:Nn \l__temp_arg_tl {,#4}
    }
    % write mathamatica code
    \iow_open:Nx \g_tmpa_iow {\g__ztikz_mma_path_tl/mma_calc_\int_use:N
    \g__mma_index_int.wls}
    \str_case:nnF {#1}{
        {part} {
            \iow_now:Nx \g_tmpa_iow {
                TeXResult = Row[DSolve[#2, #3 \l__temp_arg_tl]//Flatten, ","]/.{Rule ->
                Equal}//TeXForm//ToString;
            }
        }
        {full} {
            \iow_now:Nx \g_tmpa_iow {
                TeXResult = Row[DSolve[#2]//Flatten, ","]/.{Rule ->
                Equal}//TeXForm//ToString;
            }
        }
    }{\relax}
    % cache and input result
    \ztikz_wolfram_input_result_cs:
}
```