TikZ 坐标系统

Eureka

2023 年 11 月 24 日

目录

1 基础知识 **2**

1.1 draw 命令 2
1.2 node 命令 3
1.3 Curvels 3
1.4 positioning 库 5
1.5 路径交点 5
1.6 路径剪裁 7
1.7 支持函数 8

2 绘图库 `2d_plot` **9**

2.1 基础命令 9
2.2 函数绘制 9
2.3 阴影描绘 9
2.4 绘制精度 9
2.5 点的标注 10
2.6 交点求解 10
2.7 坐标系统绘制 11
2.8 绘图示例 11

1 基础知识

常用的`\draw`,`\fill`,`\shade`和`\filldraw`命令, 实际上都是带某些选项的`\path`命令的简写, 详情可以参见 pgfmanual 的 p172 的`actions on paths`内容.

- `\draw` $\Longleftrightarrow$ `\path[draw]`
- `\fill` $\Longleftrightarrow$ `\path[fill]`
- `\clip` $\Longleftrightarrow$ `\path[clip]`
- `\shade` $\Longleftrightarrow$ `\path[shade]`
- `\filldraw` $\Longleftrightarrow$ `\path[fill, draw]`
- `\node` $\Longleftrightarrow$ `\path node`

1.1 draw 命令

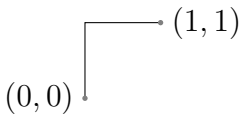
- **tikz 中坐标系的默认单位，通常是 cm; 但是在 scope 中指定偏移时一定要带上 cm 单位, 默认单位不对**
- 相对坐标: `+`，`++`的区别, `+`始终以第一个点为偏移参考点，这里每次的便宜参考点均为 $(0,1)$; 但是在`++`中，第二次的参考点即为 $(0,1)+(1,-1)$. 以下分别为两幅图的代码:

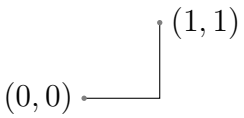
```
\draw (0,1) -- +(1,-1) -- +(1,1);
\draw (0,1) -- ++(1,-1) -- +(1,1);
```

- 极坐标: `(angle:distance)`
- 坐标运算: 需要导入`calu`库: `\usetikzlibrary{calc}`, 使用格式为:

  `\draw (0,1) -- ($(0,1)-2*(-1,1)$);`
- draw 命令的复用: 在一个`\draw`中可以同时绘制多条线段, 用法如下

  `\draw (0,0) -- (1,1) (0,1) -- (1,0);`
- draw 绘制闭合曲线: 一定要加上 `-- cycle;`
- 绘制过两点的直角折线: `\draw (0,0) |- (1,1);` 与 `\draw (0,0) -| (1,1);`





- 网格线绘制: `\draw[step=0.5] (0,0) grid (3,2);`x
- draw 绘制描点折线: `\draw plot coordinates {(0,0) (1,2) (2,1) (4,2)};`

1.2 node 命令

- node 形状: `\node[shape=rectangle]`
- node 填充: `\node[fill=red]`, 预先定义好的节点形状有三种：矩形 (rectangle), 圆形 (circle), 和点形 (coordinate)
- node 描边: `\node[draw=red]`, 默认不描边, 使用 draw=none 可以取消描边
- node 文本: `\node[text=red]`
- node 的 anchor: 默认为文本中心. CENTER
- node 文本位置: 需`positioning`库, 语法:`\node[below right=8mm and 4mm] ...;`
- node 连接点设置: 给定节点名称`\node (a) at (x, y) {};`之后,

  `a.east, a.west, a.south, a.north, a.south`

  `east, a.south west, a.north east, a.north west` 均可使用.
- coordinate: 和 node 的突然标注类似, 我们也可以使用 coordinate 进行临时标记, 语法如下: `\draw (1, 1)node[]{} -- (2, 2)coordinate(c);` 然后我们在之后便可以使用`(c)`这个点, 甚至可以这样:

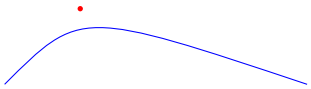
  `\draw (1, 1) -- (2, 2)coordinate(c)node[left]{node $c$};`

1.3 Curvels

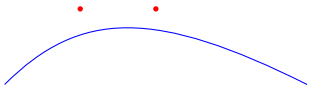
- arc:arc 命令有时是极为方便的，基本的参数中
- Bessel curvel

  此时需要将`\draw`中的`--`操作改为`..`操作, 添加曲线切线公共点，即`control`命令, 于是完整的命令为:

  ```
  \draw (0,0) .. controls (1,1) .. (4,0);
  ```



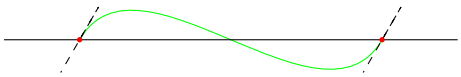
```
\draw (0,0) .. controls (1,1) and (2,1) .. (4,0);
```



- in 和 out

  使用另外一种方式, 使用 in，out两参数设置初值和结束角度:

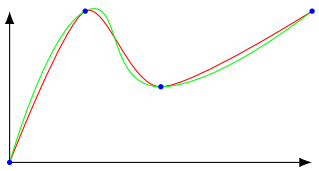
```
\draw[green] (-6, -4) to[out=60, in=-120] (-2, -4);
```



- 描点曲线

  给 plot 操作加上 smooth 选项将用光滑曲线连接各点. 而 tension 选项描述该光滑曲线的绷紧度，取值范围为从 0 到 1，默认值为 0.55.

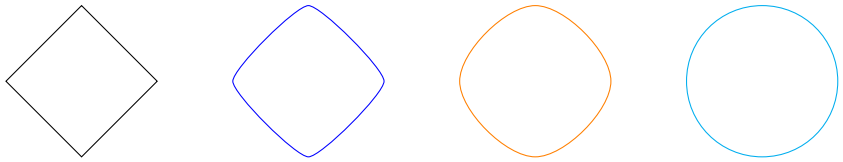
```
\draw[color=red] plot[smooth, tension=.9] coordinates ...;
```



- smooth cycle

  如果将选项 smooth 改为 smooth cycle，将绘制一条闭合曲线.

```
\begin{tikzpicture}[smooth cycle]
    \draw plot[tension=0]
        coordinates {(0,1) (1,0) (2,1) (1,2)};
    \draw[xshift=3cm] plot[tension=0.3]
        coordinates{(0,1) (1,0) (2,1) (1,2)};
    \draw[xshift=6cm] plot[tension=0.7]
        coordinates{(0,1) (1,0) (2,1) (1,2)};
    \draw[xshift=9cm] plot[tension=1]
        coordinates{(0,1) (1,0) (2,1) (1,2)};
\end{tikzpicture}
```



1.4 positioning 库

- node 文本同时指定多方向偏移量: `\node[below right=8mm and 4mm] ...;`
- node 文本偏移量可为数学表达式: `\node[below=8mm/2+4pt*2] ...;`
- node 节点之间的相对位置:`\node[fill=gray,right=1cm of a] (b) {B};`

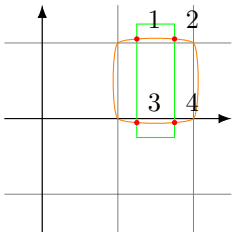
1.5 路径交点

此时你需要使用 intersections 库, `\usetikzlibrary{intersections}`, 使用语法为:

```
% Name the coordinates, but do not draw anything:
\path [name intersections={of=<path 1> and <path 2>}];
\coordinate (C) at (intersection-1);
```

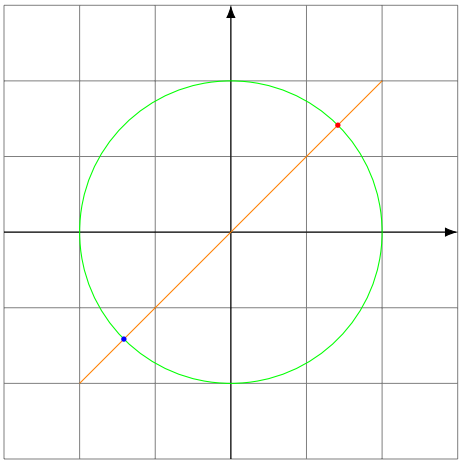
- 示例 1: 在循环中使用

```
\begin{tikzpicture}[
       every node/.style={black,above right},
       smooth cycle, scale=3
   ]
   \draw[draw=orange, name path=line 1] plot[tension=0.3]
       coordinates{(0, 0) (0,1) (1,1) (1,0)};
   \draw[draw=green, name path=line 2]
       (.25, -.25) rectangle (.75,1.25);
   \fill[red,name intersections={of=line 1 and line 2,total=\t}]
       \foreach \s in {1,...,\t}
          {(intersection-\s) circle (1pt) node {\footnotesize\s}};
\end{tikzpicture}
```



- 示例 2: 不显式调用 (不用设置变量来接受返回值), 使用`intersection-<num>` 表示第`<num>`个交点.

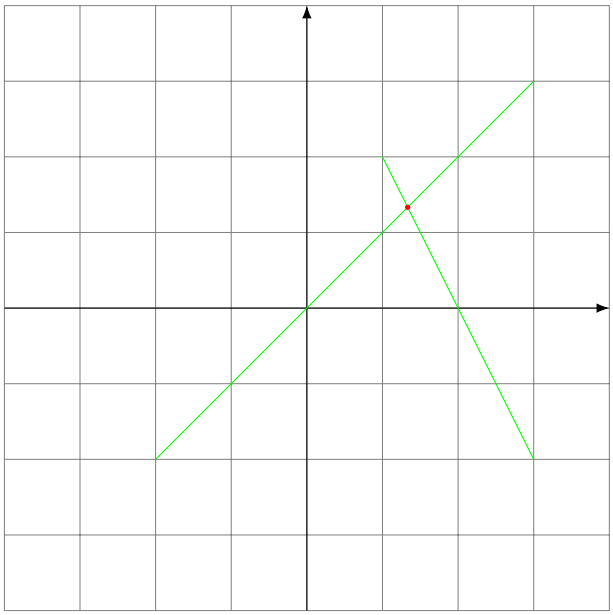
```
\begin{tikzpicture}
    \draw[orange, name path=line 1] (-2, -2) -- (2, 2);
    \draw[green, name path=circle 1] (0, 0) circle (2);
    % get the intersections(by don't get it by return value)
    \path[name intersections={of=line 1 and circle 1}];
    \fill[red] (intersection-1) circle (1pt);
    \fill[blue] (intersection-2) circle (1pt);
\end{tikzpicture}
```



- 示例 3: 不同 scope 之间共享, 此时需要把原始的

  `name path=<path name>` $\longrightarrow$ `name path global=<path name>`, 下面及一个最小示例:

```
\begin{tikzpicture}[>=Latex]
    \draw[help lines] (-4, -4) grid (4, 4);
    \draw[->] (-4, 0) -- (4, 0);
    \draw[->] (0, -4) -- (0, 4);
    \begin{scope}
        \draw[green, name path global=line 1] (-2, -2) -- (3, 3);
    \end{scope}
    \begin{scope}[xshift=2cm]
        \draw[green, name path global=line 2] (-1, 2) -- (1, -2);
    \end{scope}
    % out of scope to get the intersections
    \path[name intersections={of=line 1 and line 2}];
    \fill[red] (intersection-1) circle (1pt);
\end{tikzpicture}
```



1.6 路径剪裁

对于要使用填充功能的用户来说，路径剪裁是必须知道的，最起码得知道 `\clip`命令。不要认为这个命令很复杂，实际上你把这个 clip 裁剪的区域看作**定义域**即可，之后所有的绘图，或者是填充均只会出现在定义域，也即 clip 区域，也就是和 clip 做交集得到的区域.

1.7 支持函数

这里有一个关于函数写法注意事项：**只要你的函数表达式含有 (), 那么你的怎么函数表达式就需要使用{}来包围.**

比如下面的正确示例:

```
\draw[smooth, purple, domain=-6.927:6.927] plot (\x, {sqrt(16 - pow(\x, 2)/3)});
```

加入你写作下面这样就是错误的:

```
\draw[smooth, purple, domain=-6.927:6.927] plot (\x, sqrt(16 - pow(\x, 2)/3));
```

PGF(TiKZ) 的数学引擎支持下面这些数学函数:

abs	cot	hex	neg	rnd
acos	deg	Hex	not	round
add	depth	int	notequal	sec
and	div	ifthenelse	notgreater	sin
array	divide	less	notless	sinh
asin	e	ln	oct	sqrt
atan	equal	log10	or	subtract
atan2	factorial	log2	pi	tan
bin	false	max	pow	tanh
ceil	floor	min	rad	true
cos	frac	mod	rand	veclen
cosec	greater	Mod	random	width
cosh	height	multiply	real	

2 绘图库 2d_plot

2.1 基础命令

这里主要是依靠 Geogebra 来估计交点的坐标, 当然前面已经提到，我们可以使用自带的 intersection 库进行交点的求解. 本库的基本命令有以下函数提供:

- \NormalPlot
- \ParamPlot
- \ShowPoint
- \ShowIntersections
- \ShowGrid
- \NormalPlotPrecise
- \ParamPlotPrecise

由于 pgfplots 包太过于底层，且个人感觉没有 tikz 方便，于是本人没有对 pgfplots 宏包下太大的功夫.

2.2 函数绘制

同时这里为了解决 tikz 自身的计算能力不足问题，我自定义了两个命令:

```
\NormalPlot[<\draw cmd's option>][<variable x's domain>]{<2d function>}
\ParamPlot[<\draw cmd's option>][<param t's domain>]{<2d param function>}
```

2.3 阴影描绘

同样的也是基于上面的两个命令\NormalPlot, \ParamPlot进行阴影的绘制。此时我们有了两个新的命令

```
\NormalFill[<fill option>][<x domain>] {a set of function}
\ParamFill[<fill option>][<t domain>] {a set of param function}
```

2.4 绘制精度

并且有如下的备用选项命令\ParamPlotPrecise, \NormalPlotPrecise, 用于设置绘图的采样精度，默认在给定区间内平均采样 1000 个点, 可以使用如下命令进行这个参数的更改

```
\NormalPlotPrecise{10} % 更改NarmalPlot采样精度为10个点
\ParamPlotPrecise{10} % 更改ParamlPlot采样精度为10个点
```

注意: 改变精度命令仅对当前绘图函数的紧邻的第一个绘图命令有影响，不会影响之后的所有绘图命令的采样精度。如果想要更改之后所有绘图的采样率可以设置默认的影响范围参数once为after. 但是目前的话，只要你可选参数填的不是 once，那么它都是会改变之后的所有采样精度的. 两命令并没有用到 pgfplots 宏包axis环境.

2.5 点的标注

使用自定义的2d_plot库，可以避免自己进行如下的繁杂代码书写:

```
% 1.original plot method for parametric function
\draw[smooth, purple, domain=-6.927:6.927] plot (\x, {sqrt(16 - pow(\x, 2)/3)});
\draw[smooth, purple, domain=-6.927:6.927] plot (\x, -{sqrt(16 - pow(\x, 2)/3)});

% 2.use gnuplot gnerate data
\draw plot[smooth] file {./data.table};
\draw[red] plot[smooth] file {./param_data.table};

% 3.use pgfplots --> ticks is not the same
\begin{axis}
   \addplot gnuplot[domain=-6:6] {-sqrt(16 - x^2/3)};
\end{axis}
```

同时本绘图库还包含了基本的点的标注功能函数\ShowPoint, 使用参数如下:

```
   \ShowPoint[radius=3pt, color=blue, opacity=.5, style=circle]
      {(1, 1); (2, 2)}[text1; text2; text3][above left];
```

其中的每一个点的标注文本text_i是可以省略或者是多于欲标注的点的个数的。最后的 <style>选项为标注文本的位置, 任意的合法的\node[<option>]中的option 均可，比如[above right, font=\small]

2.6 交点求解

本绘图库包含图像 (path) 的交点求解命令\ShowIntersections, 用于绘制两条 path 的交点, 使用方法如下:

```
   \ShowIntersecions{<path 1 name>; <path 2 name>}{<num of intersections>}
```

但是目前还不推荐使用此命令, 因随之而来的就是使用自定义命令绘制的 path 进行交点的求解时可能需要处理的数据过多，导致运行速度变慢. 解决方法如下，同样是使用在线网站 Geobebra 进行交点坐标的估计. 后续也许会考虑使用 gnuplot 进行交点的计算，然后返回进行标记.

2.7 坐标系统绘制

最后便是一些关于坐标轴的基本设置函数

- `\ShowAxis`: 即展示坐标轴, 调用格式如下:

  `\ShowAxis[<axis style>]{<start>; <end>}`

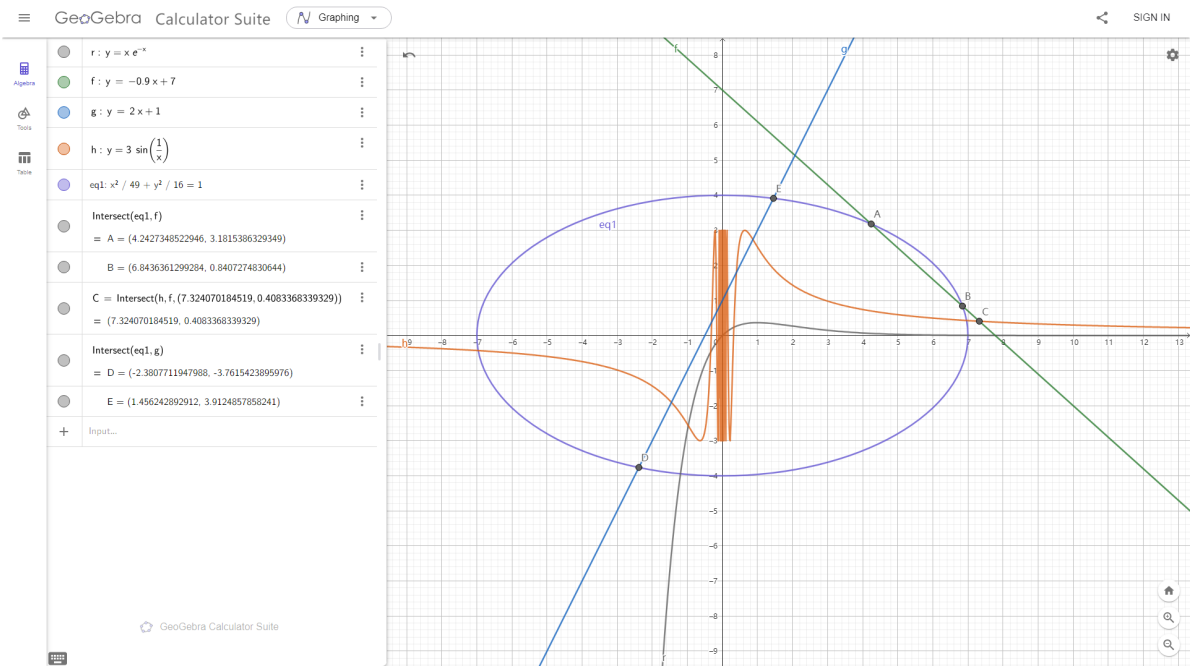
- `\ShowGrid`: 即展示坐标网格, 调用格式如下:

  `\ShowGrid[<axis style>]{<start>; <end>}`

其上的所有的`<start>`, 或`<end>`均为坐标点, 格式为`(coor1, coor2)`.
上面的所有`<style>`均为`\draw[<option>]`中的合法`<option>`字典值, 如 `color=red`.

2.8 绘图示例

首先是展示使用在线网站 Geogebra 绘制的示例, 同时也是为了展示我们的绘制精准度.



然后下面便是我们使用 `2d_plot` 绘图库的绘制效果, 经过这一点其实可以看出本绘图库的精度是可以非常高的, 取决于你设置的 gnuplot 采样精度. 另外, 如果不想要使用本库提供的`\ShowIntersections`命令显示路径交点, 那么你可以使用 Geogabra 绘制完成之后手动输入交点坐标进行绘制, 因本绘图库的坐标和 Geogebra 的坐标是 $1-1$ 的.

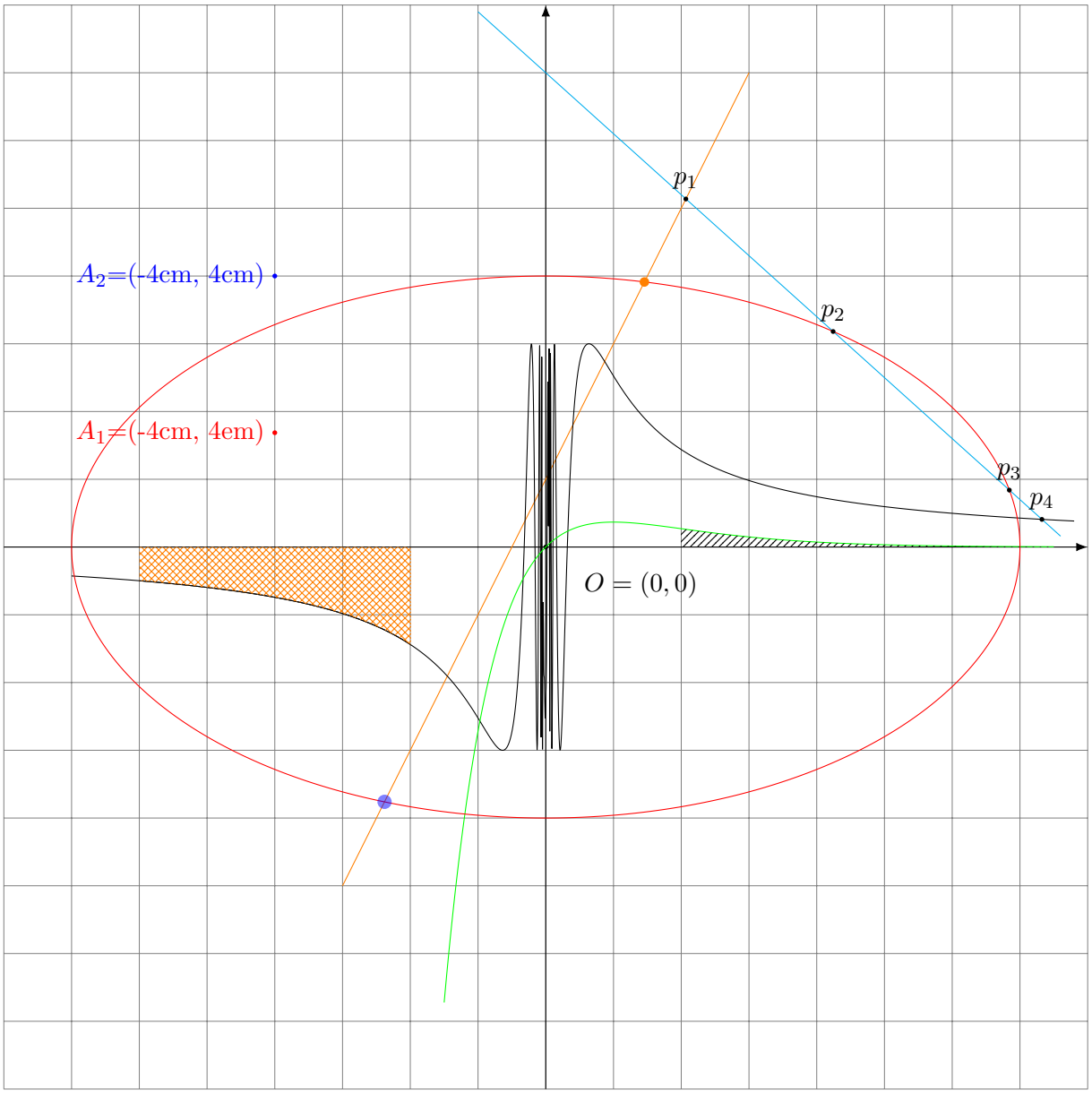


图 1: 2d_plot 绘制示例

绘图示例的代码如下:

```
\begin{tikzpicture}[font=\small, >=Latex]
    % draw grid(coordinates)
    \ShowGrid{(-8, -8); (8, 8)}
    % \fill[black] (0, 0)node[below right]{$O$=(0, 0)} circle (1pt);
    \Show{(0, 0)}[$O=(0, 0)$][below right=.5em and 1em]
    \ShowAxis {(0, -8); (0, 8)}
    \ShowAxis {(-8, 0); (8, 0)}

    % Test unit
    \fill[red] (-4cm, 4em)node[left]{$A_1$=(-4cm, 4em)} circle (1pt);
    \fill[blue] (-4cm, 4cm)node[left]{$A_2$=(-4cm, 4cm)} circle (1pt);

    % draw function
    \draw[color=orange, domain=-3:3] plot (\x, 2*\x+1);
```

```
    \draw[color=cyan, domain=-1:7.6] plot (\x, -.9*\x+7);

    % draw an ellipse
    % 1.use self define cmd
    \NormalPlotPrecise{1500}
    \NormalPlot[][-7:7.8]{3*sin(1/x)}
    \NormalPlot[green, name path=exp][-1.5:7.5] {x*exp(-x)}
    \ParamPlot[red, name path=ellipse][0:2*pi]{7*sin(t), 4*cos(t)}

    % fill(needs \usetikzlibrary{patterns})
    \begin{scope}
        \clip (2, 0) rectangle (8, 1);
        \fill[pattern=north east lines] plot
            ↪ file{./2d_plot/gnuplot_output_data/data_gen_2.table};
    \end{scope}
    \begin{scope}
        \clip (-6, 0) rectangle (-2, -2);
        \NormalPlot[][-6:-1]{3*sin(1/x)}
        \fill[pattern=crosshatch, pattern color=orange] plot
            ↪ file{./2d_plot/gnuplot_output_data/data_gen_4.table} -- (-2, 0) -- (-6, 0);
    \end{scope}

    % find intersection
    \ShowIntersecions{exp; ellipse}{2}
    % 或者是如下的语句
    % \path[name intersections={of=exp and ellipse}];
    % \ShowPoint[color=red] {(intersection-1); (intersection-2)}

    % draw intersection dot
    \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]
    \ShowPoint[radius=3pt, color=blue, opacity=.5] {(-2.380, -3.761)}
    \ShowPoint[color=orange, opacity=1, radius=2pt] {(1.456, 3.912)}
\end{tikzpicture}
```