

目录

1	Vscode 自带符号的使用	2
1.1	基本的符号	2
1.2	自带的表达式 snippets 测试	2
2	TiKZ 基础知识	2
2.1	背景知识	2
2.2	使用方法	2
2.3	直线等基本平面图形	3
2.4	贝塞尔曲线	4
2.5	绘制二维函数图像	4
2.6	draw 和 filldraw 命令	5
2.7	线条粗细	6
2.8	箭头样式	6
2.9	圆角命令	6
2.10	常见的图形变换	7
2.11	TikZ 的样式概念	7
2.12	TikZ 文字结点	8
3	TiKZ 高阶	11
3.1	综合运用	11
3.2	在 TikZ 中使用循环	11
3.3	使用循环绘制一个坐标轴	13
3.4	图形的填充	13
3.5	坐标重置	14
3.6	多图排版	14
3.7	圆柱体等三维图形的绘制	14
3.8	TiKZ 小技巧	15

1 Vscode 自带符号的使用

1.1 基本的符号

$\alpha\beta\gamma\delta\Delta\Upsilon\aleph$
♣♦♠▽★■◆▲Ⓢ
⋮⋱⋲⋳⋴⋵⋶⋷⋸⋹⋺⋻⋼⋽⋾⋿⋿
ℵℶℷℸ

1.2 自带的表达式 snippets 测试

$$\int_{i=0}^{\infty} e^x dx$$

2 TiKZ 基础知识

2.1 背景知识

- PSTricks
以 PostScript 语法为基础的绘图宏包，具有优秀的绘图能力。它对老式的 latex + dvips 编译命令支持最好，而现在的几种编译命令下使用起来都不够方便。
- TikZ & pgf
德国的 Till Tantau 教授在开发著名的 LATEX 幻灯片文档类 beamer 时一并开发了绘图宏包 pgf，目的是令其能够在 pdflatex 或 xelatex 等不同的编译命令下都能使用。TikZ 是在 pgf 基础上封装的一个宏包，采用了类似 METAPOST 的语法，提供了方便的绘图命令，绘图能力不输 PSTricks。
- METAPOST & Asymptote
METAPOST 脱胎于高德纳为 TEX 配套开发的字体生成程序 METAFONT，具有优秀的绘图能力，并能够调用 TEX 引擎向图片中插入文字和公式。Asymptote 在 METAPOST 的基础上更进一步，具有一定的类似 C 语言的编程能力，支持三维图形的绘制。
它们作为独立的程序，通常的用法是将代码写在单独的文件里，编译生成图片供 LATEX 引用，也可以借助特殊的宏包在 LATEX 代码里直接使用。

2.2 使用方法

在导言区调用 tikz 宏包，就可以用以下命令和环境使用 TikZ 的绘图功能了

```

\tikz[...] <tikz code>;

\tikz[...] {<tikz code 1>; <tikz code 2>;...}


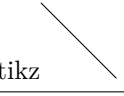
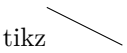
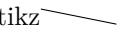

\begin{tikzpicture}[...]
<tikz code 1>;
<tikz code 2>;
...
\end{tikzpicture}

```

TikZ 用直角坐标系或者极坐标系描述点的位置。

- 直角坐标下，点的位置写作 $(\langle x \rangle, \langle y \rangle)$ ，坐标 $\langle x \rangle$ 和 $\langle y \rangle$ 可以用 L^AT_EX 支持的任意单位表示，缺省为 cm；
- 极坐标下，点的位置写作 $(\langle \theta \rangle : \langle r \rangle)$ 。 θ 为极角，单位是度。

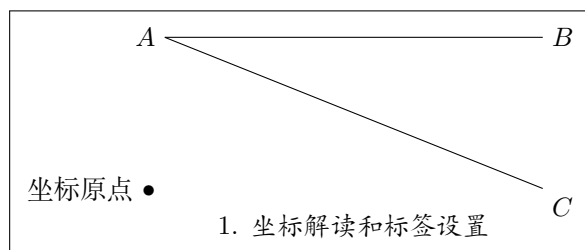
注意：绘图时的原点是相对而言的，\tikz 命令在那里，那个地方的左下角即为原点，以下即为样例

示例	样式 1	样式二
示例一		
示例二		
示例二		尺度差异

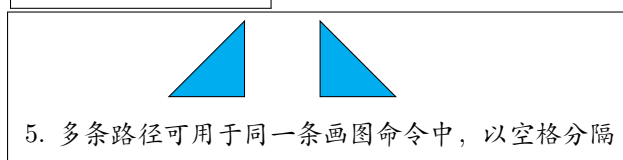
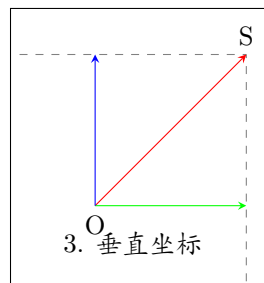
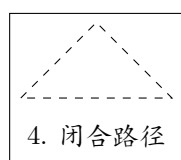
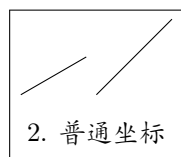
TiKZpicture 命令

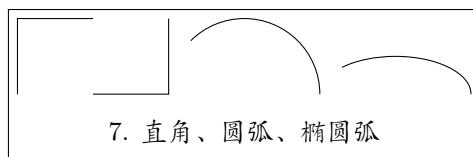
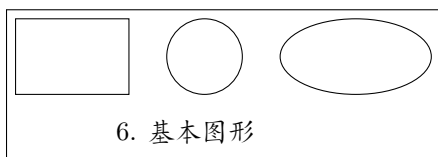
1. 语句一定要用; 结束，不然会报错；
2. 测试 tikzpicture 的排版方式 \Rightarrow 默认和文字混排

2.3 直线等基本平面图形



2. 基本图形

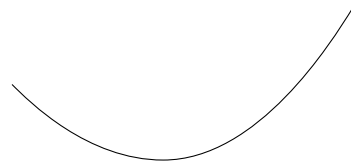




2.4 贝塞尔曲线

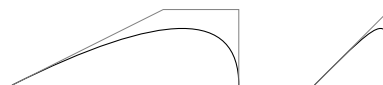
把 - 换为.. 即可. 抛物线用 parabola 操作, bend 可以指明顶点。

```
\begin{tikzpicture}
\draw (0, 1)
parabola bend (2, 0) (4.5, 2);
\node (A) at (3.8, -0.5)
{\kaishu 8. 贝塞尔曲线};
\end{tikzpicture}
```



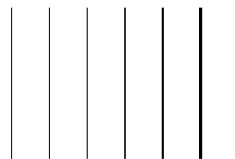
8. 贝塞尔曲线

```
\begin{tikzpicture}
\draw (0,0) .. controls
(2,1) and (3,1) .. (3,0);
\draw (4,0) .. controls
(5,1) .. (5,0);
\draw[help lines] (0,0)
-- (2,1) -- (3,1) -- (3,0)
(4,0) -- (5,1) -- (5,0);
\end{tikzpicture}
```



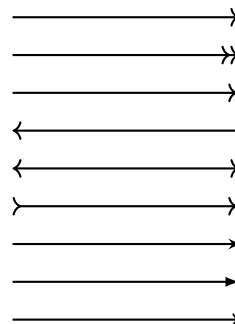
2.5 线条粗细

```
\begin{tikzpicture}
\draw[ultra thin] (0,0)--(0,2);
\draw[very thin] (0.5,0)--(0.5,2);
\draw[thin] (1,0)--(1,2);
\draw[semithick] (1.5,0)--(1.5,2);
\draw[thick] (2,0)--(2,2);
\draw[very thick] (2.5,0)--(2.5,2);
\draw[ultra thick] (3,0)--(3,2);
\end{tikzpicture}
```



2.6 箭头样式

```
\begin{tikzpicture}[thick]
  \draw[->] (0,4) -- (3,4);
  \draw[->>] (0,3.5) -- (3,3.5);
  \draw[->|] (0,3) -- (3,3);
  \draw[<-] (0,2.5) -- (3,2.5);
  \draw[<->] (0,2) -- (3,2);
  \draw[->>|] (0,1.5) -- (3,1.5);
  \draw[-stealth] (0,1) -- (3,1);
  \draw[-latex] (0,0.5) -- (3,0.5);
  \draw[-to] (0,0) -- (3,0);
\end{tikzpicture}
```



2.7 圆角命令

```
\begin{tikzpicture}
  \draw[rounded corners]
    (0,0) rectangle (1,1);
  \draw (2,0) -- (2,1)
    [rounded corners=.3cm] % .3 = 0.3
    -- (3,1) -- (3.5,0)
    [sharp corners] -- cycle;
\end{tikzpicture}
```

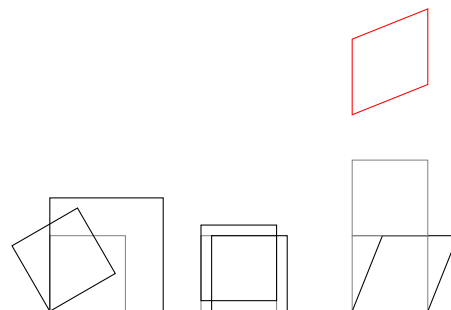


2.8 常见的图形变换

```
\begin{tikzpicture}
  % help line 参数类似于参考线
  % 变换的图形就是:后边的原始图形
  \draw[help lines](0,0) rectangle (1,1);
  \draw[scale=1.5] (0,0) rectangle (1,1);
  \draw[rotate=30] (0,0) rectangle (1,1);

  \draw[help lines](2,0) rectangle (3,1);
  \draw[yshift=4pt](2,0) rectangle (3,1);
  \draw[xshift=4pt](2,0) rectangle (3,1);

  % 图形型倾斜
  \draw[help lines](4,0) rectangle (5,1);
  \draw[xslant=0.4](4,0) rectangle (5,1);
  \draw[help lines](4,1) rectangle (5,2);
  \draw[yslant=0.4, red](4,1) rectangle (5,2);
\end{tikzpicture}
```

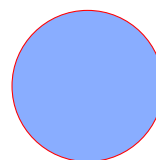


2.9 TikZ 的样式概念

1. 在环境外定义

```
% 自定义个样式,只能在最近的一个tikzpicture中使用
\tikzstyle{bluecircle}=[
    fill={rgb,255:red,137; green,173; blue,255},
    draw=red,
    shape=circle
]

% 使用自定义的样式
\begin{tikzpicture}
    \draw[bluecircle] (1, 1) circle [radius=1];
\end{tikzpicture}
```



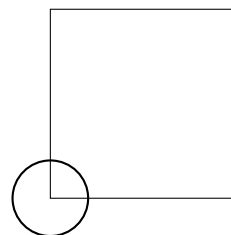
2. 在环境内定义

```
\begin{tikzpicture}
    % 定义样式
    [myarrow/.style={blue,thick,-stealth}]
    \draw (0,0)--(0,1)--(2,1);
    % 使用样式
    \draw[myarrow] (0,0)--(2,1);
    \draw[dotted] (0,0)--(2,0)--(2,1);
\end{tikzpicture}
```



3. 绘图参数或样式在局部生效

```
\begin{tikzpicture}
    \draw (0,0) rectangle (2.5, 2.5);
    % 下边这个circle就使用的是局部定义
    \begin{scope}[thick,scale=0.5]
        \draw (0,0) circle [radius=1];
    \end{scope}
\end{tikzpicture}
```



2.10 TikZ 文字结点

TikZ 用 `\node` 命令绘制文字结点:

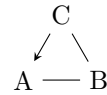
```
\node[<options>] (<name>) at (<coordinate>) {<text>};
```

`(<name>)` 为结点命名, 类似 `\coordinate`; `at (<coordinate>)` 指定结点的位置。这两者和前面的 `<options>` 都可以省略, 只有 `<text>` 是必填的。

```

\begin{tikzpicture}
\node (A) at (0,0) {A};
\node (B) at (1,0) {B};
\node (C) at (60:1) {C};
\draw[-stealth] (A) -- (B) -- (C) -- (A);
\end{tikzpicture}

```



- `anchor=<position>` 令结点的某个角落 `<position>` 与 `<coordinate>` 对应。
- `centered / above / below / left / right / above left / ... [=<length>]`
与 `anchor` 等效的选项。可选的 `<length>` 为节点相对于 `<coordinate>` 的距离。

```

\begin{tikzpicture}
\coordinate (A) at (1,1);
\fill (A) circle[radius=2pt];
% draw选项可以添加一个方框
\node[anchor=south] at (A) {a};
\node[draw,below right=4pt] at (A) {b};
\end{tikzpicture}

```



- `shape=<shape>` 结点的形状，默认可用 `rectangle` 和 `circle`，可省略 `shape=` 直接写。在导言区使用命令 `\usetikzlibrary{shapes.geometric}` 可用更多的形状。
- `text=<color>` 结点文字的颜色。
- `node font=` 结点文字的字体，形如 `\bfseries` 或 `\itshape` 等。

```

\begin{tikzpicture}
\node[
    circle,
    fill=blue,
    text=white,
    node font={\bfseries}]
(A) at (0,0) {A node};

\node[
    rectangle,
    rounded corners,
    draw=gray,
    node font={\sffamily\slshape}]
(B) at (2,0) {B node};
\end{tikzpicture}

```



inner & outer sep 参数

```

\begin{tikzpicture}
  \node () at (0, 1) {01. inner sep};
  \node[
    circle,
    draw=black,
    inner sep=10pt]
    (node) at (0,0) {1};
  \node[
    circle,
    draw=black,
    inner sep=3pt]
    (node) at (3,0) {1234};

  \node () at (0, -1) {02. outer sep};
  \node[
    circle,
    draw=blue,
    outer sep=10pt]
    (node) at (0,-2) {1};
  \node[
    circle,
    draw=blue,
    outer sep=3pt]
    (node) at (3,-2) {1};
\end{tikzpicture}

```

01. inner sep



02. outer sep




```

\begin{itemize}
  \item 测试文本,Test word:\circclenum{1}
  \item 测试文本,Test word:\circclenum{123456}
  \item 测试文本,Test word:\circclenumnew{1}
  \item 测试文本,Test word:\circclenumnew{123456}
  \item 测试文本,Test word:\mycircled{1}
  \item 测试文本,Test word:\mycircled{123456}
\end{itemize}

```

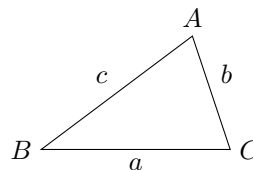
- 测试文本,Test word: ①
- 测试文本,Test word: 123456
- 测试文本,Test word: ①
- 测试文本,Test word: 123456
- 测试文本,Test word: ①
- 测试文本,Test word: 123456

标记节点和边的方法 `\node` 命令的一种等效用法是在 `\draw` 等命令的路径中使用 `node`，不仅可以对某个位置标记节点，还能够对线标记：

```

\begin{tikzpicture}
\draw (2,1.5) node[above] {$A$}
      -- node[above left] {$c$}
      (0,0) node[left] {$B$}
      -- node[below] {$a$}
      (2.5,0) node[right] {$C$}
      -- node[above right] {$b$}
      cycle;
\end{tikzpicture}

```



3 TiKZ 高阶

3.1 综合运用

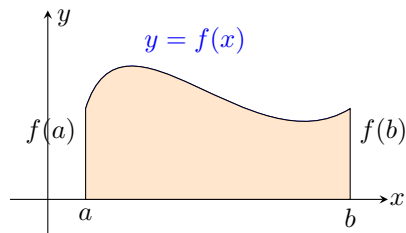
```
\begin{tikzpicture}
  \draw[-stealth] (-0.5, 0) -- (4.5, 0);
  \node[right] (xaxis) at (4.4, 0) {$x$};
  \draw[-stealth] (0, -0.5) -- (0, 2.5);
  \node[right] (yaxis) at (0, 2.4) {$y$};
  \coordinate (a) at (0.5, 0);
  \coordinate (b) at (4, 0);
  \coordinate (fa) at (0.5, 1.2);
  \coordinate (fb) at (4, 1.2);
  % 一种更简单的确定node位置的方法
  \node[below] at (a |- 0,0) {$a$};
  \node[below] at (b |- 0,0) {$b$};

  % c1, c2 为控制点
  \coordinate (c1) at (1, 2.8);
  \coordinate (c2) at (2.7, 0.4);

  \node[below left] at (fa) {$f(a)$};
  \node[below right] at (fb) {$f(b)$};
  \draw[blue]
    (fa) .. controls (c1) and (c2)
    .. node[above=3mm] {$y=f(x)$} (fb);

  % 开始填充颜色
  % \draw[fill=orange, draw=black]
  %   (a) -- (fa) -- (fb) -- (b) -- cycle;
  % 01. 填充了一个正方形出来
  % \draw[fill=orange, draw=black]
  %   (a) -- (fa) .. (fb) -- (b) -- cycle;
  % 02. --> 报错
  \draw[fill=orange!20, draw=black]
    (a) -- (fa) .. controls (c1) and (c2)
    .. (fb) -- (b) -- cycle;

  % 03. 注意: 因为路径是曲线所以你需要在中间使用 ..
  % 并且输入控制点
  % orange!20: 使用20%的orange
\end{tikzpicture}
```



3.2 在 TikZ 中使用循环

TikZ 通过 pgffor 功能宏包实现了简单的循环功能，语法为：

```
\foreach \a in {\list} {\commands}
```

上述语法定义了 `\a` 为变量，在 `{\commands}` 中使用 `\a` 完成循环。

`\list` 可以直接将所有值写出来，如 1,2,3,4；也可以写成省略形式，如 1,2,...,10。

```
\begin{tikzpicture}
\draw (0,0)--(5,0);
\foreach \i in {0.0,0.1,...,5.0}
  {\draw[very thin]
    (\i,0)--(\i,0.15);}
\foreach \I in {0,1,2,3,4,5}
  {\draw (\I,0)--(\I,0.25)
    node[above] {\I};}
\end{tikzpicture}
```



`\foreach` 还可使用**变量对**参与循环，使用 `/` 划分两个变量

```
\begin{tikzpicture}
% 这里的变量num表示第几个对象，变量var表示node的内容
\foreach \num/\var in
  {0/\alpha,1/\beta,2/\gamma}
  {\node[circle,fill=lightgray,draw]
    at (\num,0) {\var};}
\end{tikzpicture}
```

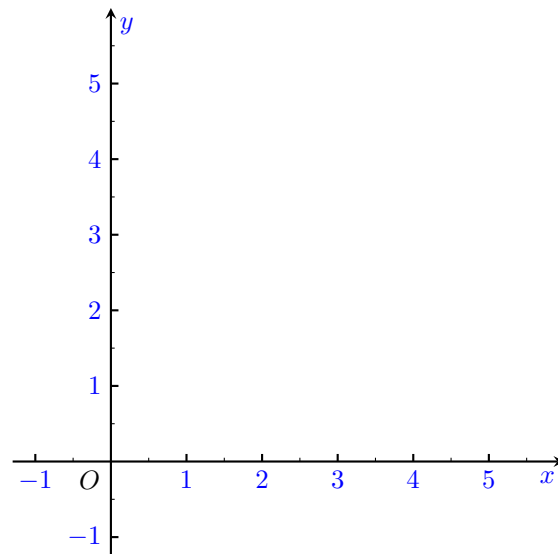


3.3 使用循环绘制一个坐标轴

```
\begin{tikzpicture}
% 1. 标记重要的点
\coordinate (O) at (0, 0);
\coordinate (ymax) at (0, 6);
\coordinate (ymin) at (0, -1.3);
\coordinate (xmax) at (6, 0);
\coordinate (xmin) at (-1.3, 0);

% 2. 绘制基本的坐标轴
\draw[-stealth, thick] (ymin) -- (ymax);
\draw[-stealth, thick] (xmin) -- (xmax);
\node[blue, right=6pt, below] at (ymax) {$y$};
\node[blue, below=6pt, left] at (xmax) {$x$};

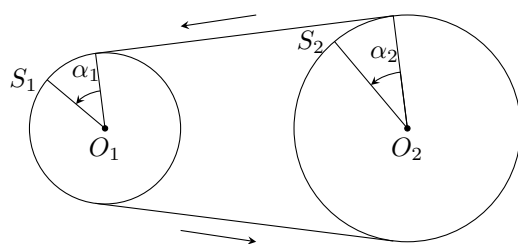
% 3. 给x, y轴加上刻度
\node[black, left=8pt, below] at (O) {$O$};
\draw[black] (O) -- (0.05, 0.5);
\draw[black] (0.5, 0) -- (0.5, 0.05);
\foreach \loc/\x in
  {-1/-1, 1/1, 2/2, 3/3, 4/4, 5/5}
  {\node[blue, below] at (\loc, 0) {$\x$};
   \draw[black, thick] (\loc, 0) -- (\loc, 0.1);
   \draw[black] (\loc+0.5, 0) -- (\loc+0.5, 0.05);
  }
\foreach \loc/\y in
  {-1/-1, 1/1, 2/2, 3/3, 4/4, 5/5}
  {\node[blue, left] at (0, \loc) {$\y$};
   \draw[black, thick] (0, \loc) -- (0.1, \loc);
   \draw[black] (0, \loc+0.5) -- (0.05, \loc+0.5);
  }
\end{tikzpicture}
```



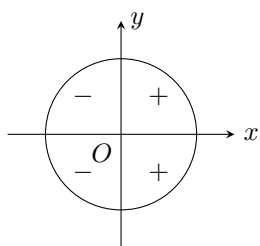
3.4 图形的填充

- 1. 类似 2.13 中的把曲线画出来：控制点已知
- 2. 类似 2.17 默认是 $(0, 0) \rightarrow f(x) \rightarrow f(x_{\max}) \rightarrow (0, 0)$ 的路径。
- 3. 把 2. 中的 $(0, 0) \rightarrow (a, b)$

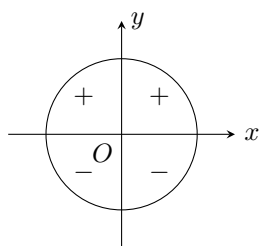
3.5 坐标重置



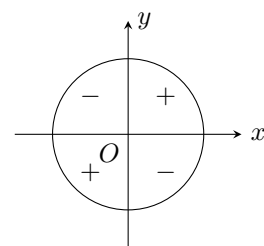
3.6 多图排版



$\cos \alpha$ 和 $\sec \alpha$



$\sin \alpha$ 和 $\csc \alpha$

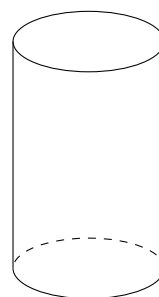


$\tan \alpha$ 和 $\cot \alpha$

3.7 圆柱体等三维图形的绘制

```
\begin{tikzpicture}
\draw (-1, 0) -- (-1, 3);
\draw (1,0) -- (1, 3);

%% 绘制椭圆
\draw (0, 3) ellipse [x radius=1, y radius=0.4];
% \draw (0, 0) ellipse [x radius=1, y radius=0.4];
\draw[dashed] (1, 0) arc [start angle=0,
                        end angle=180,
                        x radius=1,
                        y radius=0.4];
\draw (1, 0) arc [start angle=0,
                  end angle=-180,
                  x radius=1,
                  y radius=0.4];
\end{tikzpicture}
```



3.8 TiKZ 小技巧

- 1. `>=stealth`: 指定局部的范围 (tikzpicture) 的箭头样式
- 2. 填充样式的宏包: `\usetikzlibrary{patterns}`。比如 `\draw[pattern=north east lines] ...`
- 3. node 节点样式的另外一种格式: `\node (s_1) at (0, 0)[rectangle, draw=black, fill=blue]{example text}`
- 4. 三角函数默认使用度数绘图: 使用 `sin(deg(x))`, `sin(\x r)` (x 为弧度)
- 5. node 的命名 (如 s_1) 可用变量实现: `\node (s\x) at ...` (其中 \x 是 $\in \{1, 2, 3\}$ 中的变量)
- 6. bend 命令可以实现直线的弯曲 (角度) 命令: `\draw[->] to [bend left=25] ...`
- 7. 可以在 draw 的同时就把 node 给标注了 `\draw (140:1)node[left]{node1} -- (0, 0) -- (97.2:1);`