

PGF/TikZ 绘图学习笔记

zoho@bbs.ctex.org

2013 年 10 月 22 日

目录

1 开始工作	1
2 坐标表示	2
3 绘制命令	4
4 填充命令	9
5 节点命令	10
6 路径命令	12
7 文档标注	13

1 开始工作

我们先来画一些线段。例如下面的例子：

```
% 第一种方式
\tikz \draw (0,0) -- (1,1);
% 第二种方式
\tikz{\draw (0,0) -- (1,1); \draw (0,1) -- (1,0);}
% 第三种方式
\begin{tikzpicture}
\draw (0,0) -- (1,1);
\draw (0,1) -- (1,0);
\end{tikzpicture}
% 第四种方式
\tikzpicture
\draw (0,0) -- (1,1);
\draw (0,1) -- (1,0);
\endtikzpicture
```

将用四种不同的方式画出如下四个图形：



虽然我们同时使用不同方式来画图，但可以看到，基本的绘图语句就是这两句：

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

一个 TikZ 画图环境可以包含多个绘图语句，但每个语句必须以英文分号结束。否则将产生类似下面的错误提示：

```
! Package tikz Error: Giving up on this path. Did you forget a semicolon?.
```

上面这四种使用方式稍有区别，第一种方式最简单，但是它每次只能使用一个绘图语句，因为 `\tikz` 命令将在第一个分号后结束。而后面三种方式都可以包含多个绘图语句，唯一的区别在于最后一种方式可以在 Plain TeX 中使用。

2 坐标表示

再来看一下之前画线段的例子：

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



TikZ 的绘图命令大多都很直观，我们一看就知道是从坐标 (0,1) 到 (1,0) 画个线段。

在 TikZ 中，坐标的默认长度单位为厘米 (cm)，其它长度单位也可以使用，但是需要明确写出，例如：

```
\begin{tikzpicture}
\draw (0pt,30pt) -- (30pt,0pt);
\end{tikzpicture}
```



我们也可以用相对坐标，例如：

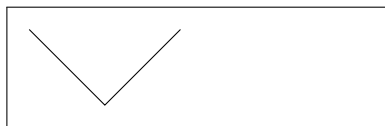
```
\begin{tikzpicture}
\draw (0,1) -- +(1,-1);
\end{tikzpicture}
```



也就是说，在坐标前加上 + 号表示相对前一个坐标作偏移。这个例子和本节第一个例子的结果是一样的。

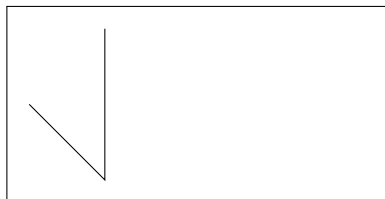
如果相对坐标需要记录下来被其它坐标使用，可以在前边用两个 + 号。例如：

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



由于第二个坐标是相对坐标，而且它又需要记录下来被第三个坐标使用，所以要用两个 + 号。作为对比，我们来看看用一个 + 号的结果：

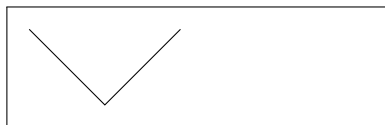
```
\begin{tikzpicture}
\draw (0,1) -- +(1,-1) -- +(1,1);
\end{tikzpicture}
```



也就是说，不管用 ++(1,-1) 还是用 +(1,-1)，第二个坐标都等于 (1,0)，但是对于前者第三个坐标等同于 (2,1)，对于后者第三个坐标等同于 (1,2)。

当然，TikZ 中也可以用极坐标，此时角度值和长度值之间用冒号隔开。例如：

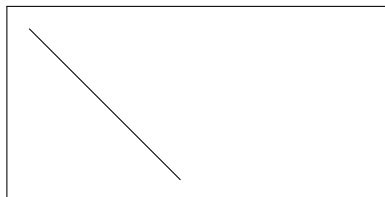
```
\begin{tikzpicture}
\draw (90:1) -- (0:1) -- (2,1);
\end{tikzpicture}
```



可以看到，直角坐标和极坐标混合使用是没有问题的。

如果需要对坐标进行运算，可以在导言区用 `\usetikzlibrary{calc}` 命令载入 `calc` 扩展，然后用类似下面的例子：

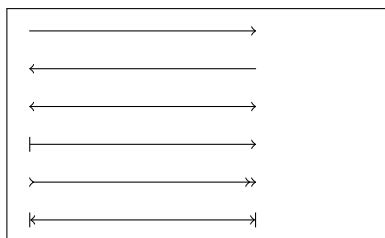
```
\begin{tikzpicture}
\draw (0,1) -- ($ (0,1) - 2 * (-1,1) $);
\end{tikzpicture}
```



3 绘制命令

到目前为止，我们知道用 `\draw` 命令可以画线段。类似的我们也可以画带箭头的线段。例如：

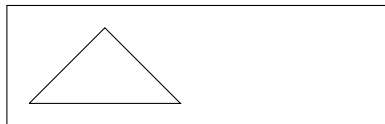
```
\begin{tikzpicture}
\draw[->] (0,2.5) -- (3,2.5);
\draw[<-] (0,2) -- (3,2);
\draw[<->] (0,1.5) -- (3,1.5);
\draw[|>] (0,1) -- (3,1);
\draw[>->>] (0,0.5) -- (3,0.5);
\draw[|<->|] (0,0) -- (3,0);
\end{tikzpicture}
```



和其它 LaTeX 命令一样，TikZ 命令的可选参数也放在方括号中。利用带箭头的线段，画坐标系就毫无问题了。

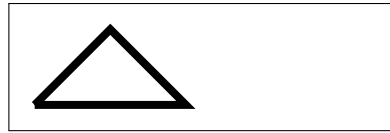
既然我们已经知道怎么画线段，而 TikZ 中多个操作是可以连续指明的，这样画三角形也没问题了。例如：

```
\begin{tikzpicture}
\draw (0,0) -- (1,1) -- (2,0) -- (0,0);
\end{tikzpicture}
```



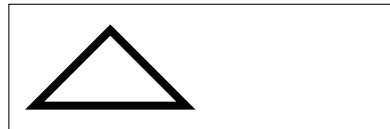
TikZ 默认的线条宽度为 0.4pt。我们可以用 `line width` 选项来加粗这个三角形：

```
\begin{tikzpicture}[line width=3pt]
\draw (0,0) -- (1,1) -- (2,0) -- (0,0);
\end{tikzpicture}
```



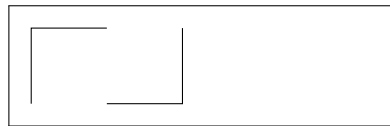
这时候，问题出来了：我们发现，加粗三角形的左下角有个缺口！这个问题可以用 `cycle` 操作来解决。把前面的例子按如下的方法修改，一切就都正常了：

```
\begin{tikzpicture}[line width=3pt]
\draw (0,0) -- (1,1) -- (2,0) -- cycle;
\end{tikzpicture}
```



再来看怎么画过两点的直角折线：

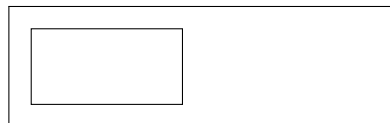
```
\begin{tikzpicture}
\draw (0,0) |- (1,1);
\draw (1,0) -| (2,1);
\end{tikzpicture}
```



其中 `|-` 表示先竖直画线再水平画线，而 `-|` 正好反过来。

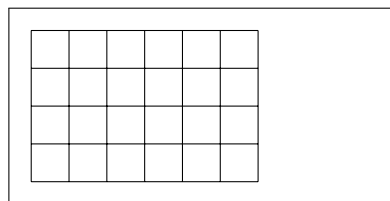
将两个直角折线合起来就得到一个矩形。不过我们有更简洁的写法：

```
\begin{tikzpicture}
\draw (0,0) rectangle (2,1);
\end{tikzpicture}
```



类似地，我们也可以画网格线。例如：

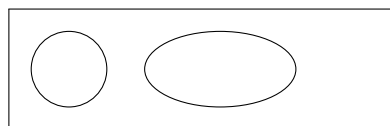
```
\begin{tikzpicture}
\draw[step=0.5] (0,0) grid (3,2);
\end{tikzpicture}
```



其中的 `step` 参数指明网格的间隙。

不能总是画最简单的线段，我们接下来来画一种二次曲线：圆和椭圆。例如：

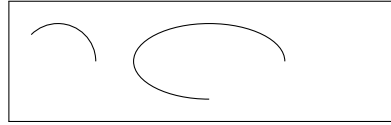
```
\begin{tikzpicture}
\draw (0,0) circle (0.5);
\draw (2,0) ellipse (1 and 0.5);
\end{tikzpicture}
```



对于圆，我们需要圆心和半径这两个参数，而对于椭圆，我们需要中心，长轴和短轴这三个参数。

当然，我们也可以画圆弧和椭圆弧。此时，需要将圆心或中心换成起始点，再指明起始角度和终结角度。例如：

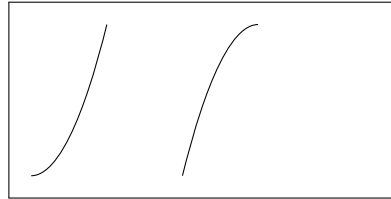
```
\begin{tikzpicture}
\draw (0.5,0) arc (0: 135: 0.5);
\draw (3,0) arc (0: 270: 1 and 0.5);
\end{tikzpicture}
```



注意后面的参数涉及到角度，所以它们之间用冒号分隔。

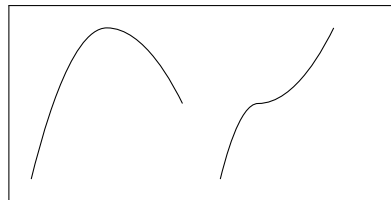
抛物线也不在话下，只需要指明从起始点和终结点坐标。例如：

```
\begin{tikzpicture}
\draw (0,0) parabola (1,2);
\draw (2,0) parabola[bend at end] (3,2);
\end{tikzpicture}
```



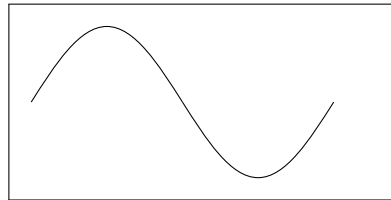
TikZ 默认以起始点为抛物线顶点；如果加上 **bend at end** 选项，则以终结点为顶点。我们也可以在中间指明顶点坐标，此时将得到由两段抛物线连接而成的曲线。例如：

```
\begin{tikzpicture}
\draw (0,0) parabola bend (1,2) (2,1);
\draw (2.5,0) parabola bend (3,1) (4,2);
\end{tikzpicture}
```



正弦和余弦曲线也可以类似地画出来。例如：

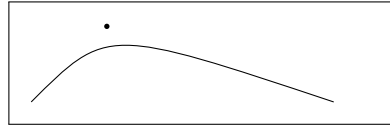
```
\begin{tikzpicture}
\draw (0,0) sin (1,1) cos (2,0);
\draw (2,0) sin (3,-1) cos (4,0);
\end{tikzpicture}
```



其中，对于 $(0,0) \sin (1,1)$ 操作，TikZ 拿 $[0, \frac{\pi}{2}]$ 区间上的正弦曲线作适当伸缩得到，而对于 $(2,0) \sin (3,-1)$ 操作，除了作伸缩外，还需要再旋转 180 度才能得到。余弦的绘制过程类似。我们将两者结合起来，就能画出一个周期的正弦或者余弦曲线。

我们还可以画出贝塞尔（Bézier）曲线。此时需要将 **--** 操作改为 **..** 操作。例如：

```
\begin{tikzpicture}
\draw (0,0) .. controls (1,1) .. (4,0);
\fill (1,1) circle (1pt);
\end{tikzpicture}
```



这样画出的是二次贝塞尔曲线，其中曲线在起点 $(0,0)$ 和终点 $(4,0)$ 的切线都过点 $(1,1)$ ，因此点 $(1,1)$ 称为控制点。代码中的 `\fill` 是填充命令，这里用于画一个实心小圆点，我们后面还会再看到这个命令。

如果提供两个控制点，就可以画出三次贝塞尔曲线。例如：

```
\begin{tikzpicture}
\draw (0,0) .. controls (1,1) and (2,1) .. (4,0);
\fill (1,1) circle (1pt) (2,1) circle (1pt);
\end{tikzpicture}
```



此时，曲线在起点 $(0,0)$ 的切线过第一个控制点 $(1,1)$ ，而在终点 $(4,0)$ 的切线过第二个控制点 $(2,1)$ 。

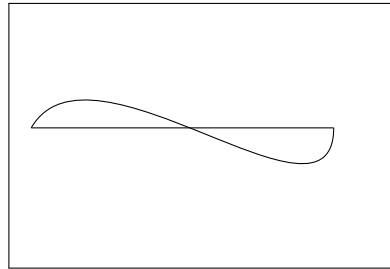
三次贝塞尔曲线的形状可以更加复杂，例如下面的例子：

```
\begin{tikzpicture}
\draw (0,0) .. controls (1,1) and (2,1) .. (4,1);
\fill (1,1) circle (1pt) (2,1) circle (1pt);
\end{tikzpicture}
```



在绘制连接两点的曲线时，我们也可以指明起始角度和终结角度。例如：

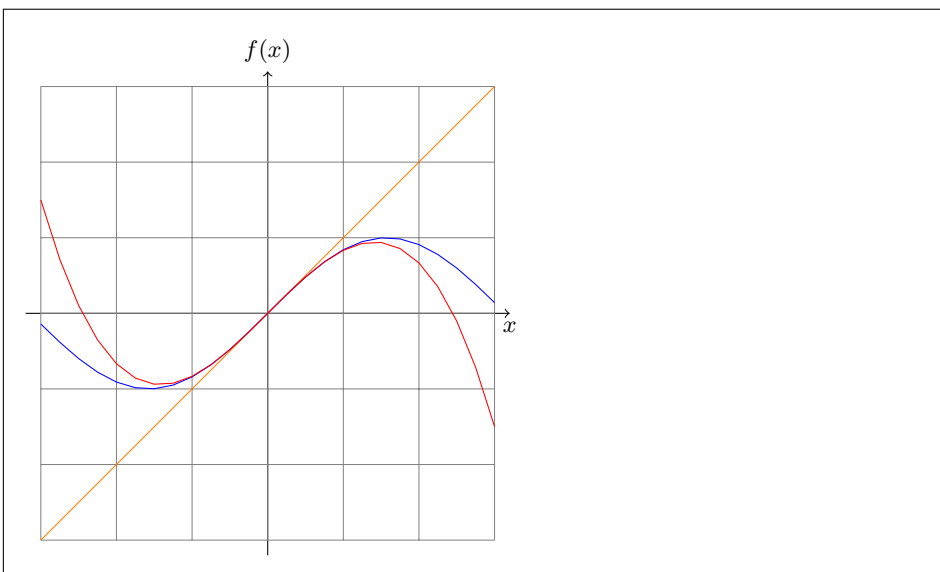
```
\begin{tikzpicture}
\draw (0,0) to[out=60,in=-90] (4,0);
\draw (0,0) to (4,0);
\end{tikzpicture}
```



如果 `to` 没有任何选项，就和前面的 `--` 操作同样画出线段。实际上，`to` 操作是比较一般的操作，它还有多种选项，这里不再详述。

最后，我们还可以用 `plot` 操作来画一般的函数曲线。例如：

```
\begin{tikzpicture}[domain=-3:3]
\draw[->] (-3.2,0) -- (3.2,0) node[below] {$x$};
\draw[->] (0,-3.2) -- (0,3.2) node[above] {$f(x)$};
\draw[very thin,color=gray] (-3,-3) grid (3,3);
\draw[color=orange] plot (\x,\x);
\draw[color=blue] plot (\x,{sin(\x r)});
\draw[color=red] plot (\x,{\x-(1/6)*(\x)^3});
\end{tikzpicture}
```



其中的 `node` 节点命令后面会介绍。注意代码中的 `sin` 函数需要指明长度类型 `r` (表示弧度，默认为角度)。

4 填充命令

现在来看填充命令 `\fill`。先来看如何画实心圆盘和实心矩形。例如：

```
\begin{tikzpicture}
\fill (0.5,0.5) circle (0.5);
\fill[blue] (2,0) rectangle (4,1);
\end{tikzpicture}
```



默认的填充颜色是黑色，当然我们可以自己设定颜色。

注意 `\draw` 仅绘制区域边界，而 `\fill` 仅填充区域内部，不包括边界。两者结合我们可以绘制边界和内部颜色不同的圆盘或者矩形。例如：

```
\begin{tikzpicture}
\draw[green] (0,0) rectangle (3,1);
\fill[orange] (0,0) rectangle (3,1);
\end{tikzpicture}
```

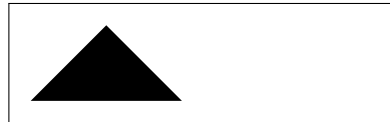


或者，我们可以将 `\draw` 和 `\fill` 合并起来写成 `\filldraw` 命令。此时，这个例子就可以改为

```
\begin{tikzpicture}
\filldraw[fill=orange,draw=green] (0,0) rectangle (3,1);
\end{tikzpicture}
```

我们也可以填充自己绘制的封闭曲线，例如：

```
\begin{tikzpicture}
\fill (0,0) -- (1,1) -- (2,0) -- cycle;
\end{tikzpicture}
```



与 `\fill` 命令类似的还有 `\shade` 这个渐变命令。例如：

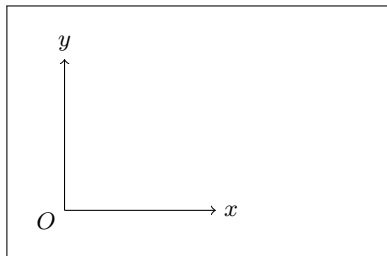
```
\begin{tikzpicture}
\shade[inner color=yellow,outer color=orange] (1,1) circle (1);
\shade[left color=gray,right color=black] (3,0) rectangle (6,2);
\end{tikzpicture}
```



5 节点命令

如果想在图形上加上标签，就要用到节点命令 `\node` 了。例如下面的例子绘制了直角坐标系：

```
\begin{tikzpicture}
\draw[->] (0,0) -- (2,0);
\draw[->] (0,0) -- (0,2);
\node[below=4pt,left] at (0,0) {$0$};
\node[right] at (2,0) {$x$};
\node[above] at (0,2) {$y$};
\end{tikzpicture}
```



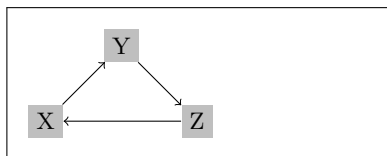
节点命令的一般形式如下：

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

其中 `<text>` 是节点的文本，`<coordinate>` 是节点的坐标，`<name>` 是节点的名称，而 `<options>` 是一些选项。在前面的例子里，`left`，`right`，`above` 和 `below` 分别表示节点文本相对于节点坐标的四种位置。倘若不指明位置，节点文本的中心将和节点的坐标重合。

至于节点名称的用法，可以先看下面这个例子：

```
\begin{tikzpicture}
\node[fill=lightgray] (a) at (0,0) {X};
\node[fill=lightgray] (b) at (1,1) {Y};
\node[fill=lightgray] (c) at (2,0) {Z};
\draw[->] (a) -- (b);
\draw[->] (b) -- (c);
\draw[->] (c) -- (a);
\end{tikzpicture}
```



可以看到，知道了节点名称，就可以把它看成坐标那样，作各种线段和曲线的连接。这就是示意图或者流程图的一般绘制方法。

节点之间也可以用相对位置。这时候，我们首先需要载入 `positioning` 库：

```
\usetikzlibrary{positioning}
```

然后就可以用类似下面例子的方法使用相对位置了：

```
\begin{tikzpicture}
\node[fill=gray] (a) {X};
\node[fill=gray,right=1 of a] (b) {Y};
\node[fill=gray,right=1 of b] (c) {Z};
\draw[->] (a) -- (b);
\draw[->] (b) -- (c);
\end{tikzpicture}
```



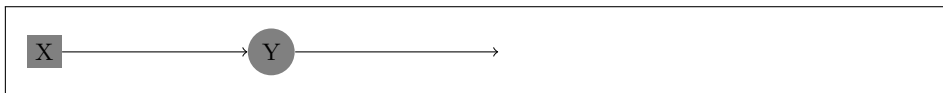
节点命令和 `\draw` 等绘制命令实际上是可以写在一起的。例如本节最开始画坐标轴的例子就可以改写如下：

```
\begin{tikzpicture}
\draw[->] (0,0) node[below=4pt,left]{$0$} -- (2,0) node[right]{$x$};
\draw[->] (0,0) -- (0,2) node[above]{$y$};
\end{tikzpicture}
```

这样写可能简短一点，但是对于初学者来说有些复杂。

TikZ 中的节点形状预先定义好的有三种：矩形，圆形和点形，但可以自己定义新的形状。节点形状用 `shape` 选项来确定（不指定则默认为矩形），例如：

```
\begin{tikzpicture}
\node[shape=rectangle,fill=gray] (a) at (0,0) {X};
\node[shape=circle,fill=gray] (b) at (3,0) {Y};
\node[shape=coordinate,fill=gray] (c) at (6,0) {Z};
\draw[->] (a) -- (b);
\draw[->] (b) -- (c);
\end{tikzpicture}
```



可以看到，由于点形节点的面积为零，所以填充颜色和标注文本都没有任何效果。

这种点形节点还可以用简化的记号来表示。例如上面例子的点形节点也可以这样表示：

```
\coordinate (c) at (6,0);
```

6 路径命令

之前介绍的 `\draw`, `\fill`, `\shade` 和 `\filldraw` 命令, 实际上, 分别是 `\path[draw]`, `\path[fill]`, `\path[shade]` 和 `\path[fill,draw]` 命令的简写。而 `\node` 和 `\coordinate` 命令分别等同于 `\path node` 和 `\path coordinate`。

例如, 下面这四种写法的效果是一样的, 它们都画出一个半径为 1cm 的圆:

```
\draw (0,0) circle (1cm);
\path [draw] (0,0) circle (1cm);
\path (0,0) [draw] circle (1cm);
\path (0,0) circle (1cm) [draw];
```

而下面这两种写法也是一样的, 它们都定义了一个节点:

```
\node [fill=gray] (a) at (1,1) {X};
\path (1,1) node[fill=gray] (a) {X};
```

TikZ 的路径命令, 将之前介绍的多种命令统一起来了。我们来整理一下。首先, 路径命令的一般形式如下:

```
\path <specification>;
```

即路径命令后面需要写上路径描述 `<specification>`, 然后以分号结尾。而路径描述可以有多种, 这些我们都介绍过了:

- `(coordinate) (coordinate)`: 从一个点移动到另一个点
- `(coordinate) -- (coordinate)`: 从一个点画线段到另一个点
- `(coordinate) rectangle (coordinate)`: 以这两个点为对角画矩形
- `(coordinate) circle[options]`: 以这个点为圆心画圆
- `(coordinate) arc[options]`: 以这个点为起点画圆弧
- `(coordinate) .. controls (control) and (control) .. (coordinate)`: 从一个点画贝塞尔曲线到另一个点
- `(coordinate) to[options] (coordinate)`: 从一个点按照选项指定画某种曲线到另一个点
- `(coordinate) node[options] (name) {text}`: 将这个点记为文本节点
- `(coordinate) coordinate (name)`: 将这个点记为点形节点

7 文档标注

利用 TikZ 的节点命令，我们也可以对文档标注。基本方法先标出节点，然后再画图。首先我们定义一个 `\tikzmark` 命令如下：

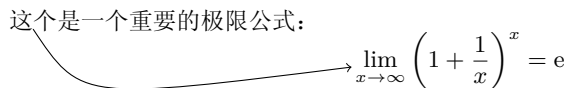
```
\newcommand\tikzmark[1]{%
  \tikz[overlay,remember picture] \node[coordinate] (#1) {};%
}
```

其中行尾的 `%` 号用于去掉后面的多余空格。

现在就可以用这个命令来标注了。例如：

```
这\tikzmark{a}个是一个重要的极限公式：
\[ \tikzmark{b} \lim_{x \rightarrow \infty} \left( 1 + \frac{1}{x} \right)^x = \mathrm{e} \]
\begin{tikzpicture}[overlay,remember picture]
\draw[->] (a) .. controls +(2em,-3em) .. (b);
\end{tikzpicture}
```

这个是一个重要的极限公式：



$$\lim_{x \rightarrow \infty} \left(1 + \frac{1}{x} \right)^x = e$$

注意在标出节点和添加标注这两个步骤中，`overlay,remember picture` 选项都是必需的。这种标注方法需要编译两遍才能看到最终的结果。

在制作 Beamer 演示时，这种标注的方法会更加有用。实际上，我们也可以直接使用 `tikzmark` 这个功能更加强大的文档标注包¹。

¹<http://www.ctan.org/pkg/tikzmark>