LaTeX 2$_\varepsilon$ Tutorial

记录我的 *LaTeX* 学习过程

笔　　者： 黄正阳

邮　　箱： huangzy@nwafu.edu.cn

GitHub： https://github.com/huangzy1218

目录

目录 i

绘图功能 iii

0.1 LaTeX 绘图简介 iii

0.2 TikZ 绘图语言 iii

0.2.1 绘图方式 iii

0.2.2 TikZ坐标 iv

0.2.3 TikZ路径 v

0.2.4 TikZ绘图绘图命令和参数 vii

0.2.5 文字节点 ix

0.2.6 绘制曲线 xi

源代码示例列表

0.1 遗传算法 xii

0.2 二叉树前序遍历 xiii

绘图功能

0.1 LaTeX 绘图简介

LaTeX 提供了原始的 picture 环境，能够绘制一些基本的图形如点、线、矩形、圆、Bézier 曲线等等，不过受制于 LaTeX 本身，它的绘图功能极为有限，效果也不够美观。

当前主流的 LaTeX 绘图宏包/程序主要有：

- PSTricks

  以 PostSciprt 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。

- Ti*k*Z & 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。Ti*k*Z 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。

0.2 Ti*k*Z 绘图语言

0.2.1 绘图方式

在导言区引入 tikz 宏包，就可使用命令或环境进行绘图操作。

- 命令模式

  `\tikz[...]` ⟨*tikz code*⟩`;`

- 命令分组模式

  `\tikz[...] {`⟨*tikz code 1*⟩`;` ⟨*tikz code 2*⟩`; ...}`

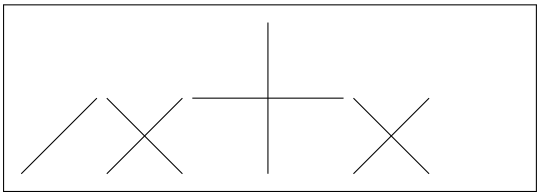
- 环境模式

```
\begin{tikzpicture}[...]
⟨tikz code 1⟩;
⟨tikz code 2⟩;
...
\end{tikzpicture}
```

- 起止命令模式

```
\tikzpicture
⟨tikz code 1⟩;
⟨tikz code 2⟩;
...
\endtikzpicture
```

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



0.2.2 Ti*k*Z坐标

Ti*k*Z 用直角坐标系或者极坐标系描述点的位置。

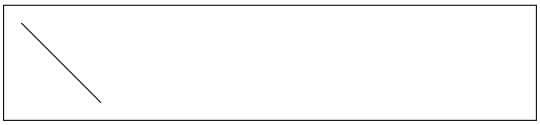
- 直角坐标下，点的位置写作 (⟨x⟩,⟨y⟩)，坐标 ⟨x⟩ 和 ⟨y⟩ 可以用 LaTeX 支持的任意单位表示，缺省为 cm；
- 极坐标下，点的位置写作 (⟨θ⟩:⟨r⟩)。θ 为极角，单位是度。

使用绝对坐标

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

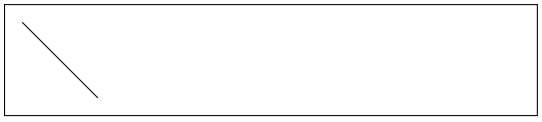
使用坐标单位，默认为 cm，此时为绝对坐标

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



使用相对坐标，即相对第一个坐标的偏移

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



记录相对坐标，使用 ++

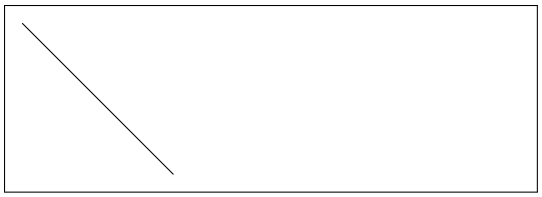
```
\begin{tikzpicture}
\draw (0,1) -- ++(1,-1) -- +(1,1);
\end{tikzpicture}
```

使用极坐标，单位为 °

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

使用\usetikzlibrary{calc} 载入 calc 扩展，可对坐标进行计算：

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

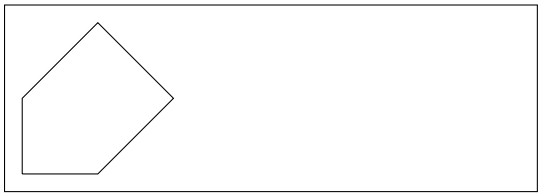


0.2.3 TikZ路径

线段和折线

若需要连续绘制，可连续使用–：

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



我们还可以为某个点命名：`\coordinate (A) at (⟨coordinate⟩)` 然后就可以使用 `(A)` 作为点的位置了。

```
\begin{tikzpicture}
\draw (0,0) -- (30:1);
\draw (1,0) -- (2,1);
\coordinate (S) at (0,1);
\draw (S) -- (1,1);
\end{tikzpicture}
```

坐标的表示形式还包括“垂足”形式：

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

设置环境的默认参数[line width]可改变线宽：

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

由上例发现图形存在缺口，可通过 cycle 命令令路径回到起点，生成闭合的路径。

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

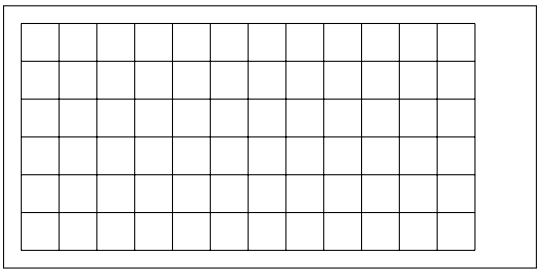
其它常用的路径还包括：

- 矩形

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

- 网格、函数图像，网格可用 step 参数控制网格大小，函数图像用 domain 参数控制定义域：

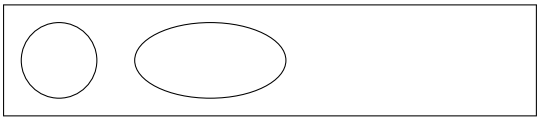
```
\begin{tikzpicture}
% step指明网格间隔
\draw[step=0.5](0,0) grid (6,3);
\end{tikzpicture}
```



曲线

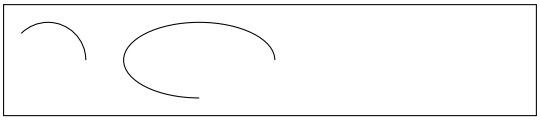
绘制圆时，需要指定圆心和半径参数；绘制椭圆时，则需要指定圆心，并通过 ellipse 指定长轴和短轴：

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



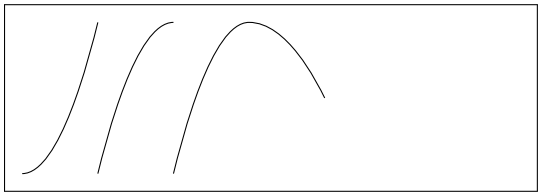
使用 arc 绘制圆弧和椭圆弧：

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



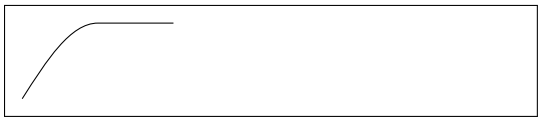
使用 parabola 绘制抛物线，默认起始点为抛物线顶点，设置可选参数 bend at end 选项以终结点为顶点。

```
\begin{tikzpicture}
\draw (0,0) parabola (1,2);
\draw (1,0) parabola[bend at end]
(2,2);
% 指定中间顶点坐标
\draw (2,0) parabola bend
(3,2) (4,1);
\end{tikzpicture}
```



使用 sin 和 cos 绘制三角函数曲线：

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

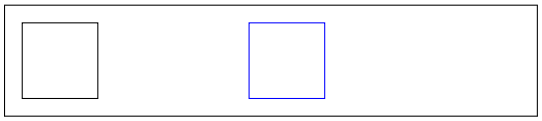


0.2.4 TikZ绘图绘图命令和参数

除了 \draw 命令之外，TikZ 还提供了 \fill 命令用来填充图形，\filldraw 命令则同时填充和描边。以下示例常用的一些绘图命令和参数：

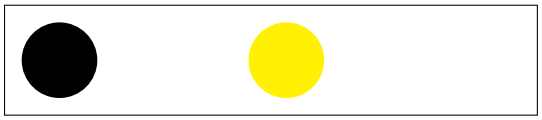
- \draw[color]

```
\begin{tikzpicture}
\draw (0,0) rectangle (1,1);
\draw[blue] (3,0) rectangle (4,1);
\end{tikzpicture}
```



- \fill[color]

```
\begin{tikzpicture}
\fill (0,0.5) circle (0.5);
\fill[yellow] (3,0.5) circle[radius=0.5];
\end{tikzpicture}
```

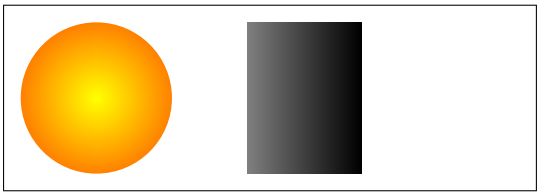


- \filldraw[fill=color,draw=color]

```
\begin{tikzpicture}
\filldraw[fill=orange,draw=red]
(0,0) rectangle (4.5,1);
\end{tikzpicture}
```

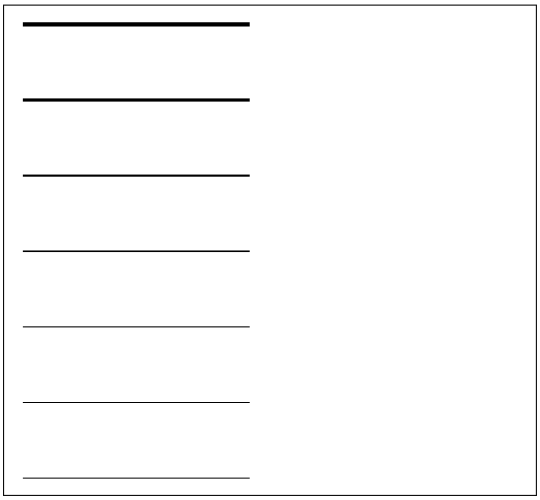
- \shade[inner color=color,outer color=color]
  \shade[left color=color,right color=color]

```
\begin{tikzpicture}
\shade[inner color=yellow,
outer color=orange]
(1,1) circle (1);
\shade[left color=gray,
right color=black]
(3,0) rectangle (4.5,2);
\end{tikzpicture}
```



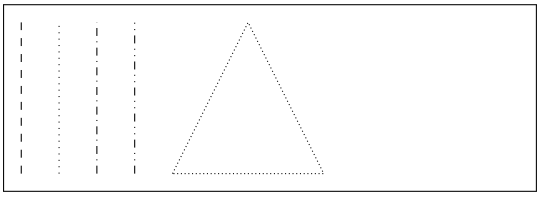
- `thick=`⟨*length*⟩`/thin/semithick/...` 指定线条的粗细：

```
\begin{tikzpicture}
\draw[ultra thin] (0,0)--(3,0);
\draw[very thin] (0,1)--(3,1);
\draw[thin] (0,2)--(3,2);
\draw[semithick] (0,3)--(3,3);
\draw[thick] (0,4)--(3,4);
\draw[very thick] (0,5)--(3,5);
\draw[ultra thick] (0,6)-- (3,6);
\end{tikzpicture}
```



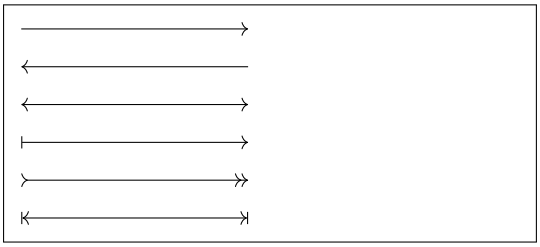
- `solid/dashed/dotted/dash dot/dash dot dot` 指定线条类型（实线、虚线、点划线等）。与 `dashed` 对应地有 `densely dashed` 和 `loosely dashed`，后三种类型同理。

```
\begin{tikzpicture}
\draw[dashed] (0,0) -- (0,2);
\draw[dotted] (0.5,0) -- (0.5,2);
\draw[dash dot] (1,0) -- (1,2);
\draw[dash dot dot] (1.5,0) -- (1.5,2);
\draw[densely dotted]
(2,0) -- (3,2) -- (4,0) -- cycle;
\end{tikzpicture}
```



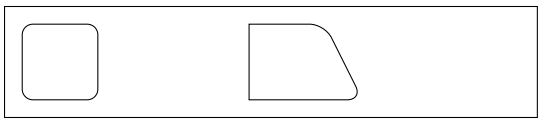
- ⟨*arrow*⟩`-`⟨*arrow*⟩ 指定线条首尾的箭头形式。复杂的箭头形式需要在导言区使用 `\usetikzlibrary {arrows.meta}`。

```
\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}
```



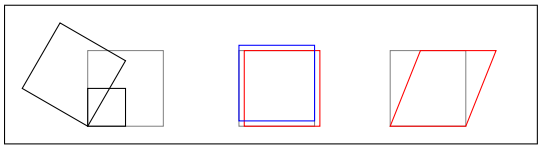
- `rounded corners`[`=`⟨*radius*⟩]`/sharp corners` 将路径转向处绘制成圆角/直角。可选参数⟨*radius*⟩控制圆角的半径。可以对某一段路径直接使用。

```
\begin{tikzpicture}
\draw[rounded corners]
(0,0) rectangle (1,1);
\draw (3,0)--(3,1)
[rounded corners=.2cm]
--(4,1)--(4.5,0)
[sharp corners]--cycle;
\end{tikzpicture}
```



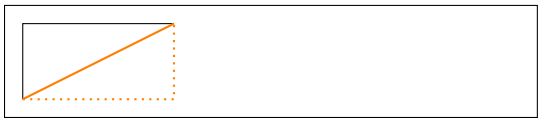
- scale/xshift/yshift/xslant/yslant/rotate 设定图形的缩放、位移和旋转。

```
\begin{tikzpicture}
\draw[help lines] (0,0) rectangle (1,1);
\draw[scale=0.5] rectangle (1,1);
\draw[rotate=60] rectangle (1,1);
\draw[help lines] (2,0) rectangle (3,1);
\draw[xshift=2pt,color=red]
(2,0) rectangle (3,1);
\draw[yshift=2pt,color=blue]
(2,0) rectangle (3,1);
\draw[help lines](4,0) rectangle (5,1);
\draw[xslant=0.4,color=red]
(4,0) rectangle (5,1);
\end{tikzpicture}
```



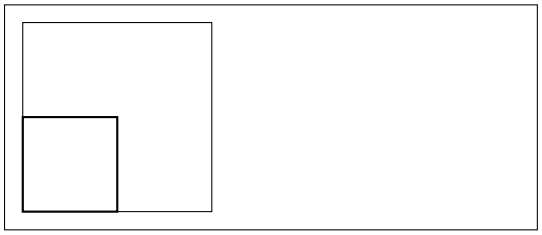
为了重复利用绘图参数，减少代码冗余，TikZ 引入了“样式”的概念，可以定义一个样式包含绘图参数，然后将样式作为一个参数用于绘图：

```
\begin{tikzpicture}
[myarrow/.style={orange,thick}]
\draw (0,0)--(0,1)--(2,1);
\draw[myarrow] (0,0)--(2,1);
\draw[myarrow,dotted]
(0,0)--(2,0)--(2,1);
\end{tikzpicture}
```



TikZ 还提供了 scope 环境，令绘图参数或样式在局部生效：

```
\begin{tikzpicture}
\draw (0,0) rectangle (2.5, 2.5);
\begin{scope}[thick,scale=0.5]
\draw (0,0) rectangle (2.5, 2.5);
\end{scope}
\end{tikzpicture}
```



0.2.5 文字节点

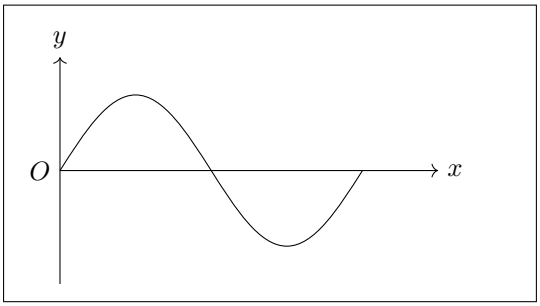
使用\node 命令，基本格式如下：

\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。

⟨*options*⟩ 可选择 `left`、`right`、`above`、`below`，若不指明位置，则结点文本中心与其结点坐标重合。

```
\begin{tikzpicture}
 \draw[->] (0,0)--(5,0);
 \draw[->] (0,-1.5)--(0,1.5);
 \node[below,left] at (0,0) {$O$};
 \node[right] at (5,0) {$x$};
 \node[above] at (0,1.5) {$y$};
 \draw (0,0) sin (1,1) cos (2,0)
  sin (3,-1) cos (4,0) ;
\end{tikzpicture}
```




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

`\node` 可指定特定的参数：

- `[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*⟩ 为结点相对于 ⟨*coordinate*⟩ 的距离。

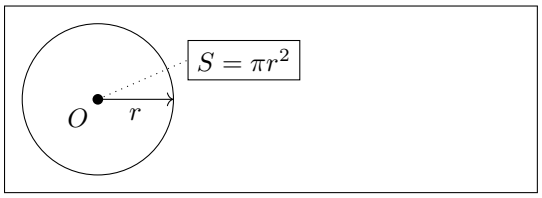
```
\begin{tikzpicture}
\coordinate (A) at (1,1);
\fill (A) circle[radius=2pt];
\node[draw,anchor=west] at (A) {a};
\node[draw,above left] at (A) {b};
\end{tikzpicture}
```

- `[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={`⟨*font command*⟩`}]` 结点文字的字体，形如 `\bfseries` 或 `\itshape` 等。

```
\begin{tikzpicture}
\node[circle,fill=black,text=white]
(A) at (0,0) {1};
\node[rectangle,rounded corners,
color=white,fill=red] (B)
at (3,0) {a new node};
\draw[->] (A)--(B);
\end{tikzpicture}
```

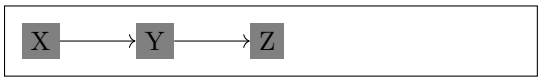
- [inner sep=⟨*length*⟩ / outer sep=⟨*length*⟩] 结点边界向外和向内的额外距离。
- [minimum size=⟨*length*⟩ / minimum height=⟨*length*⟩ / minimum width=⟨*length*⟩] 结点的最小大小或最小高度/宽度。

```
\begin{tikzpicture}
\draw (0,0) circle[radius=1];
\fill (0,0) circle[radius=2pt];
\node[draw] (P) at (15:2)
{$S = \pi r^2$};
\draw[->] (0,0)--(1,0);
\node[below] at (0.5, 0) {$r$};
\node[below left] at (0,0) {$O$};
\draw[dotted] (0,0) -- (P.west);
\end{tikzpicture}
```



节点之间使用相对位置，需引入 kzlibrarypositioning 库。

```
\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}
```



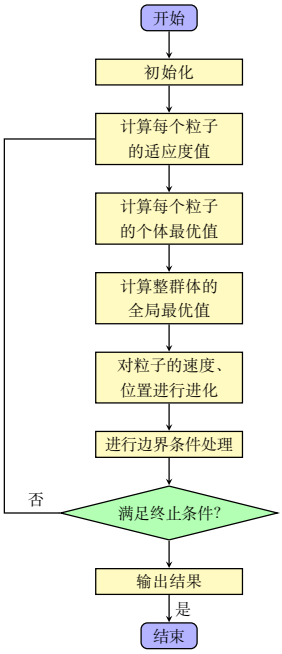
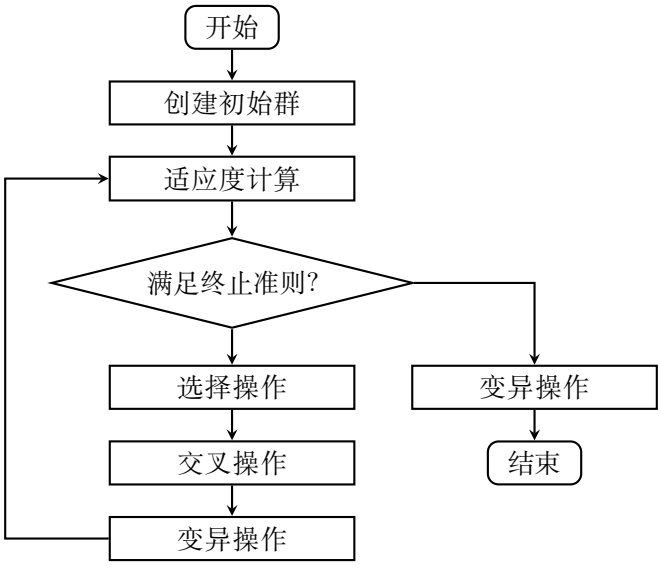


图 0.2.1: 粒子群算法流程图

0.2.6 绘制曲线

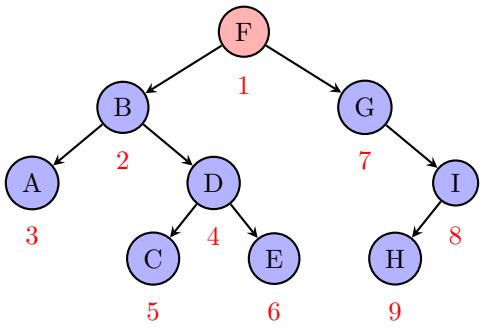
plot 操作绘制平面曲线，默认为描点连线：

```
\begin{tikzpicture}[node distance=1cm]
% 定义流程图具体形状
 \tikzstyle{startstop} = [rectangle, rounded corners,thick,
 minimum width=1cm, minimum height=0.5cm,text centered,
 draw=black, fill=white,text width=1cm] % 开始
 \tikzstyle{process} = [rectangle,thick, minimum width=3cm,
 minimum height=0.5cm, text centered,
 draw=black, fill=white,text width=3cm]        % 步骤
 \tikzstyle{decision} = [diamond,aspect = 4, thick, minimum width=3cm,
 minimum height=0.5cm, text centered, draw=black, fill=white]% 判断框
 \tikzstyle{arrow} = [thick,->,>=stealth]        % 箭头
 \node (start) [startstop] {开始};
 \node (pro1) [process, below of=start] {创建初始群};
 \node (pro2) [process, below of=pro1] {适应度计算};
 \node (dec1) [decision, below of=pro2,yshift=-0.38cm] {满足终止准则？};
 \node (pro3) [process, below of=dec1,yshift=-0.38cm] {选择操作};
 \node (pro4) [process, below of=pro3] {交叉操作};
 \node (pro5) [process, below of=pro4] {变异操作};
 \node (pro6) [process, right of=pro3,xshift=3cm] {变异操作};
 \node (stop) [startstop, below of=pro6] {结束};

 % 连接具体形状
 \draw [arrow](start) -- (pro1);
 \draw [arrow](pro1) -- (pro2);
 \draw [arrow](pro2) -- (dec1);
 \draw [arrow](dec1) -- (pro3);
 \draw [arrow](pro3) -- (pro4);
 \draw [arrow](pro4) -- (pro5);
 \draw [arrow](pro5) -| (-3,-2) |- (pro2);
 \draw [arrow](dec1) -| (pro6);
 \draw [arrow](pro6) -- (stop);
\end{tikzpicture}
```



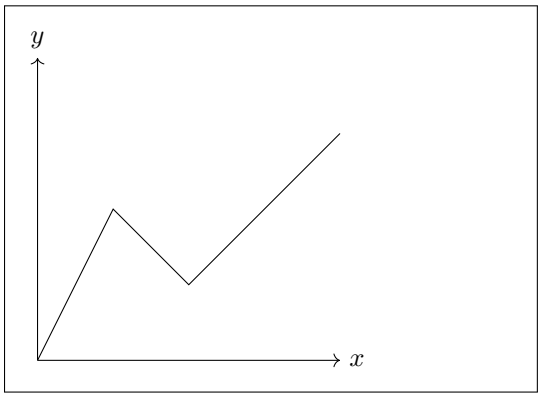
源代码 0.1: 遗传算法

```
\begin{tikzpicture}

\end{tikzpicture}
```



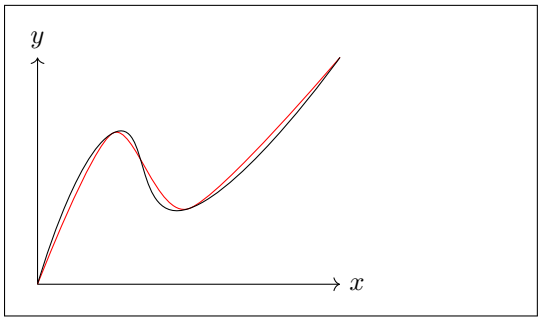
源代码 0.2: 二叉树前序遍历

```
\begin{tikzpicture}
\draw[->](0,0)--(4,0)node[right]{$x$};
\draw[->](0,0)--(0,4)node[above]{$y$};
% 画一般的平面曲线（描点连线）
\draw plot coordinates
{(0,0)(1,2)(2,1)(4,3)};
\end{tikzpicture}
```

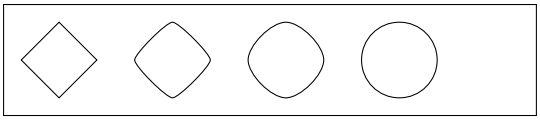


若需要绘制光滑曲线，可使用[smooth,tension]参数，其中 tension 指定紧绷度，取值范围从 0 到 1，默认为 0.55：

```
\begin{tikzpicture}
  \draw[->](0,0)--(4,0)node[right]{$x$};
  \draw[->](0,0)--(0,3)node[above]{$y$};
  \draw[color=red] plot[smooth]
  coordinates {(0,0) (1,2) (2,1) (4,3)};
  \draw plot[smooth,tension=.9]
  coordinates {(0,0) (1,2) (2,1) (4,3)};
\end{tikzpicture}
```

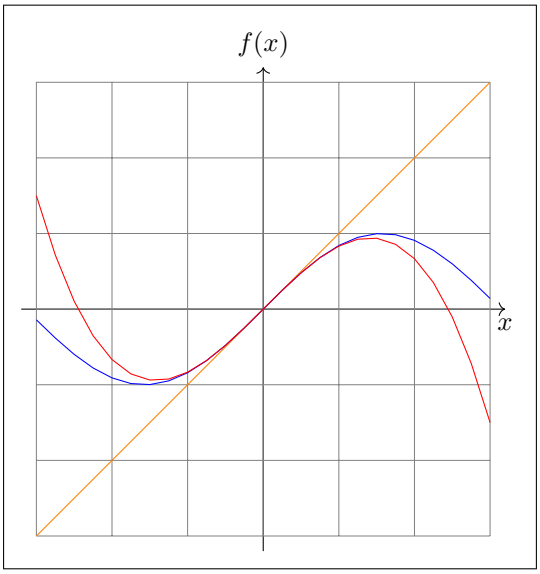


```
\begin{tikzpicture}[smooth cycle]
 \draw plot[tension=0] coordinates
  {(0,0.5) (0.5,0) (1,0.5) (0.5,1)};
% tension=0 将画出正方形,
% 而tension=1 将画出圆形
 \draw[xshift=1.5cm] plot[tension=0.3]
  coordinates
  {(0,.5) (.5,0) (1,.5) (.5,1)};
 \draw[xshift=3cm] plot[tension=0.7]
  coordinates
  {(0,.5) (.5,0) (1,.5) (.5,1)};
 \draw[xshift=4.5cm] plot[tension=1]
  coordinates
  {(0,.5) (.5,0) (1,.5) (.5,1)};
\end{tikzpicture}
```



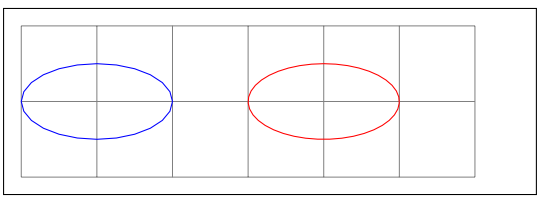
一般函数曲线通过 plot\x,\y 绘制。

```
\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}
```



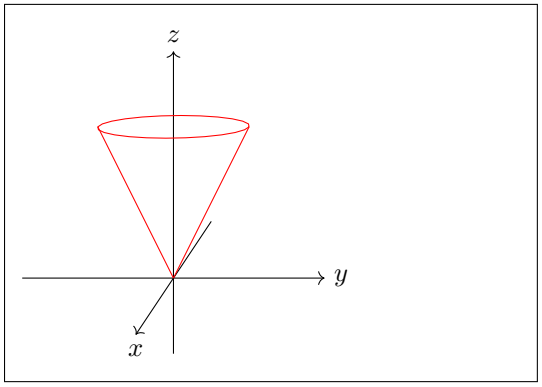
参数曲线在参数后添加\r，样点数 samples 决定了曲线的光滑程度。

```
\begin{tikzpicture}[domain=0:2*pi]
 \draw[very thin,color=gray](-2,-1)
      grid (4,1);
 \draw[color=blue,xshift=-1cm]plot
      ({sin(\x r)},{0.5*cos(\x r)});
 \draw[color=red,xshift=2cm,samples=40]
      plot({sin(\x r)},{0.5*cos(\x r)});
\end{tikzpicture}
```



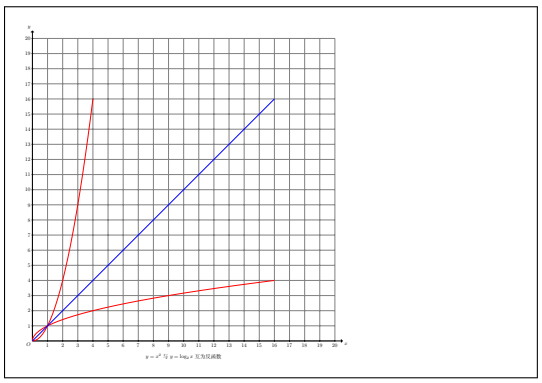
绘制简单圆锥面，需要指明三个坐标各自的单位向量的位置。

```
\begin{tikzpicture}[x={(-.1cm,-.15cm)},
 y={(1cm,0cm)},z={(0cm,1cm)}]
 \draw[->](-5,0,0)--(5,0,0)
  node[below]{$x$};
 \draw[->](0,-2,0)--(0,2,0)
  node[right]{$y$};
 \draw[->](0,0,-1)--(0,0,3)
  node[above]{$z$};
 \draw[color=red] plot[domain=0:2*pi]
  ({sin(\x r)},{cos(\x r)},2);
  \draw[color=red] (0,0,0)--(0,1,2)
   (0,0,0)--(0,-1,2);
\end{tikzpicture}
```



绘制网格坐标：

```
\scalebox{0.2}{
\begin{tikzpicture}
 \draw[help lines] (0,0) grid (20,20);
 \draw [->] (0,0)--(20.5,0)
  node[below right] {$x$};
 \draw [->] (0,0)--(0,20.5)
  node[above left] {$y$};
 \node[below left] at (0,0) {$O$};
 \foreach \i in {1,...,20}
 \draw (\i,-0.05)--++(90:0.1)
  node[below=1mm]{\i};
 \foreach \i in {1,...,20}
 \draw (0.05,\i)--++(180:0.1)
  node[left=-0.5mm]{\i};
  \draw[red,very thick]
   plot[domain=0:4](\x,{(\x)^2});
  \draw[red,very thick]
   plot[domain=0:4]({(\x)^2},\x);
  \draw[help lines,blue]
   plot[domain=0:16](\x,\x);
  \node[below of=2cm] at (10,0)
  {$y=x^2$与$y=\log_{2}x$互为反函数};
\end{tikzpicture}}
```

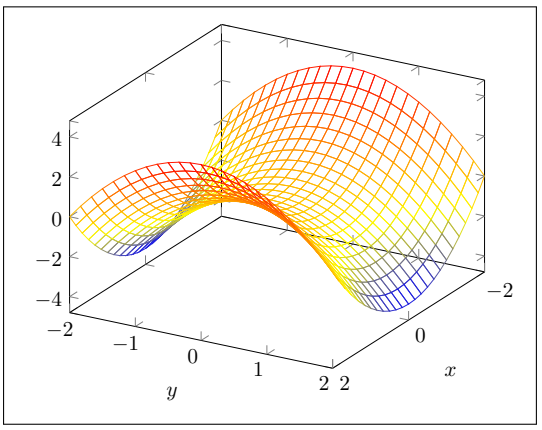


pgfplots 是一个可视化工具，可以简化文档中的绘图。基本思想是提供输入数据公式，而 pgfplots 完成其余工作。其基本格式如下：

```
\addplot3[...]coordiant{point1, point2, ...}
```

- 用 mesh 选项绘制网格曲面

```
\scalebox{0.8}{
 \begin{tikzpicture}
  \begin{axis}[view={120}{30},
   xlabel=$x$,ylabel=$y$]
   \addplot3
    [domain=-2:2,y domain=-2:2,mesh]
    {x^2-y^2};
  \end{axis}
 \end{tikzpicture}}
```



- 用 surf 选项绘制网格曲面

```
\scalebox{0.8}{
\begin{tikzpicture}
\begin{axis}[view={120}{30},
xlabel=$x$,ylabel=$y$]
\addplot3
[domain=-2:2,y domain=-2:2,surf]
{x^2-y^2};
\end{axis}
\end{tikzpicture}}
```