使用 TikZ 几何作图

2025 年 3 月 19 日

目录

第 1 章　基本作图命令

这里的命令都是通过 /tikz/insert path[1] 在当前路径上插入新的路径, 或者是通过 `def` 组合一些 tikz 命令.

1.1　坐标轴 Axes

调用方式

```
\axes (xmin:xmain,ymin:ymax);
```

参数说明

xmin,xmax x 范围

ymin,ymax y 范围

示例

见1.2.

1.2　坐标变换 Coordinates Transformations

调用方式

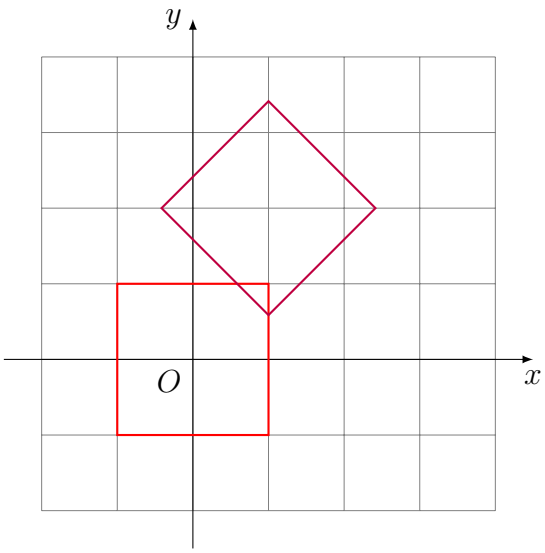
```
transform={angle:(xshift,yshift)}
```

参数说明

绕原点旋转 angle, 然后水平方向和竖直方向分别平移 xshift 和 yshift

示例

```
\begin{tikzpicture}
  % \draw[help lines] (-5,-5) grid (5,5);
  % \draw[-latex] (-5.5,0) -- (5.5,0) node[below] {$x$};
  % \draw[-latex] (0,-5.5) -- (0,5.5) node[left] {$y$};
  % \draw (0,0) node[below left] {$O$};
  \axes (-2:4, -2:4);
  \draw[thick,red] (-1,-1) rectangle (1,1);
  \draw[thick,purple,transform={45:(1,2)}] (-1,-1)
  ↪  rectangle (1,1);
\end{tikzpicture}
```



1.3 仿射组合 Affine Combination

调用方式

```
affine={A,B,k}
```

参数说明

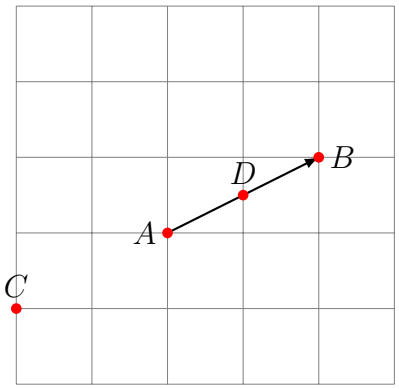
A, B 两点坐标

k 系数

返回点 A, B 的仿射组合: $A + k \cdot (B - A)$.

示例

```
\begin{tikzpicture}
  \draw[help lines] (0,0) grid (5,5);
  \coordinate (A) at (2,2);
  \coordinate (B) at (4,3);
  \coordinate [affine={A,B,-1}] (C);
  \coordinate [affine={A,B,.5}] (D);
  \draw[thick, -latex] (A) -- (B);
  \foreach \p/\placement in {A/left,B/right,
  C/above,D/above}{
    \fill[red] (\p) circle (2pt);
    \draw (\p) node[\placement] {$\p$};
  }
\end{tikzpicture}
```



1.4 中点 Midpoint

调用方式

```
midpoint={A,B}
```

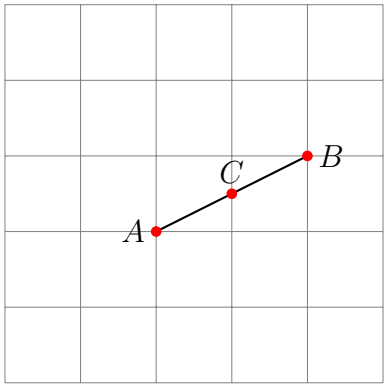
参数说明

A, B 两点坐标

返回点 A, B 的中点坐标.

示例

```
\begin{tikzpicture}
  \draw[help lines] (0,0) grid (5,5);
  \coordinate (A) at (2,2);
  \coordinate (B) at (4,3);
  \coordinate [midpoint={A,B}] (C);
  \draw[thick] (A) -- (B);
  \foreach \p/\placement in {A/left,B/right,
  C/above}{
    \fill[red] (\p) circle (2pt);
    \draw (\p) node[\placement] {$\p$};
  }
\end{tikzpicture}
```



1.5 平移 Translate

调用方式

```
translate={A,B,C}
```

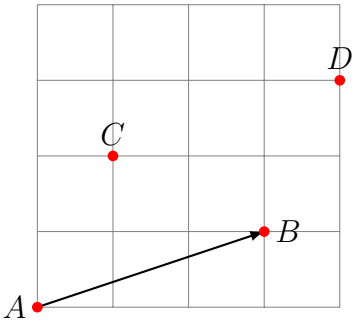
参数说明

A,B,C 三点坐标

返回 C 按向量 AB 移动所得的坐标: $C+(B-A)$.

示例

```
\begin{tikzpicture}
  \draw[help lines] (0,0) grid (4,4);
  \coordinate (A) at (0,0);
  \coordinate (B) at (3,1);
  \coordinate (C) at (1,2);
  \coordinate [translate={A,B,C}] (D);
  \draw[thick, -latex] (A) -- (B);
  \foreach \p/\placement in
  ↪ {A/left,B/right,C/above,D/above}{
    \fill[red] (\p) circle (2pt);
    \draw (\p) node[\placement] {$\p$};
  }
\end{tikzpicture}
```



1.6 对称点 Reflect

调用方式

```
reflect={A,B,C}
```

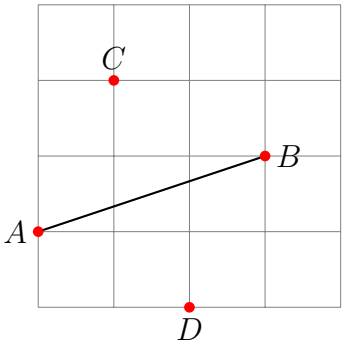
参数说明

A,B,C 三点坐标

返回 C 关于直线 AB 的对称点的坐标 (设 D 为 C 在 AB 的投影): $C+2(D-C)$.

示例

```
\begin{tikzpicture}
  \draw[help lines] (0,0) grid (4,4);
  \coordinate (A) at (0,1);
  \coordinate (B) at (3,2);
  \coordinate (C) at (1,3);
  \coordinate [reflect={A,B,C}] (D);
  \draw[thick] (A) -- (B);
  \foreach \p/\placement in
  ↪  {A/left,B/right,C/above,D/below}{
    \fill[red] (\p) circle (2pt);
    \draw (\p) node[\placement] {$\p$};
  }
\end{tikzpicture}
```



1.7 投影 Project

调用方式

```
project={A,B,C}
```

参数说明

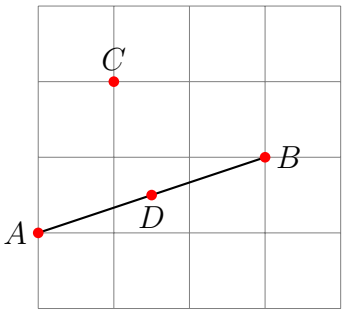
A,B,C 三点坐标

返回 C 在直线 AB 的投影的坐标.

示例

```
\begin{tikzpicture}
  \draw[help lines] (0,0) grid (4,4);
  \coordinate (A) at (0,1);
  \coordinate (B) at (3,2);
  \coordinate (C) at (1,3);
  \coordinate [project={A,B,C}] (D);
  \draw[thick] (A) -- (B);
  \foreach \p/\placement in
  ↪  {A/left,B/right,C/above,D/below}{
    \fill[red] (\p) circle (2pt);
    \draw (\p) node[\placement] {$\p$};
```

```
  }
\end{tikzpicture}
```



1.8 反演 Inverse

调用方式

```
circle inverse={O,A,P}
```

参数说明

O 圆心

A 圆上一点

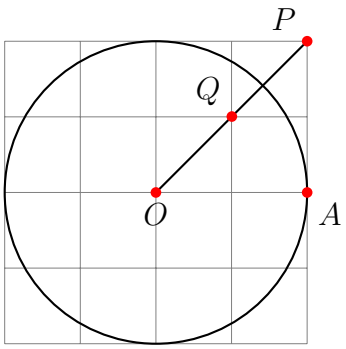
P 平面上任一点

返回 P 关于圆 (O, A) 的反演点.

示例

```
\begin{tikzpicture}
  \draw[help lines] (-2,-2) grid (2,2);
  \coordinate (O) at (0,0);
  \coordinate (A) at +(0:2);  % 圆上一点，相对坐标
  \coordinate (P) at (2,2);
  \coordinate[circle inverse={O,A,P}] (Q);
```

```
    \draw[thick,circle={O,A}];
    \draw[thick] (O) -- (P);

    \foreach \p/\placement in {O/below,A/below right,
    P/above left,Q/above left}{
      \fill[red] (\p) circle (2pt);
      \draw (\p) node[\placement] {$\p$};
    }
  \end{tikzpicture}
```



1.9 旋转 Revolve

调用方式

```
revolve={A,B}
```

参数说明

A,B 两点坐标

注 为了避免覆盖 tikz 的 rotate, 这里将旋转命令为 revolve.

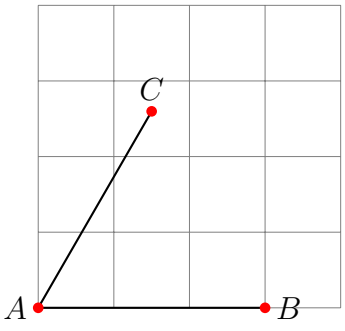
返回 B 绕 A 旋转的点.

还需要指定 revolve/angle (default: 0) 和 revolve/angle scale(default: 1) 两个选项, 可以通过下面的方式来指定 /revolve/angle:

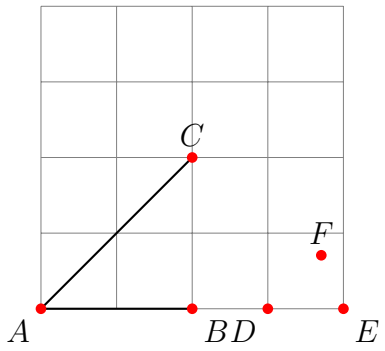
1. 直接指定角度: `revolve/angle=60`
2. 位置向量与 x 轴夹角: `revolve/angle={P1}`
3. 两位置向量的夹角: `revolve/angle={P1,P2}`
4. 由三点定义的角 (P_1 为顶点, P_2 为起点, P_3 为终点): `revolve/angle={P1,P2,P3}`
5. 两向量的夹角 (逆时针方向): `revolve/angle={P1,P2,P3,P4}`

示例

```
\begin{tikzpicture}
  \draw[help lines] (0,0) grid (4,4);
  \coordinate (A) at (0,0);
  \coordinate (B) at (3,0);
  \coordinate [revolve/angle=60, revolve={A,B}] (C);
  \draw[thick] (A) -- (B) (A) -- (C);
  \foreach \p/\placement in {A/left,B/right,C/above}{
    \fill[red] (\p) circle (2pt);
    \draw (\p) node[\placement] {$\p$};
  }
\end{tikzpicture}
```

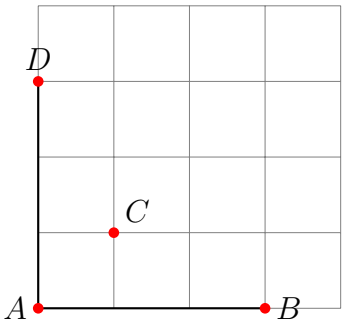


```
\begin{tikzpicture}
  \draw[help lines] (0,0) grid (4,4);
  \coordinate (A) at (0,0);
  \coordinate (B) at (2,0);
  \coordinate (C) at (2,2);
  \coordinate (D) at (3,0);
  \coordinate (E) at (4,0);
  \coordinate [revolve/angle={A,B,C},revolve={D,E}] (F);
  \draw[thick] (A) -- (B) (A) -- (C);
  \foreach \p/\placement in {
  A/below left,B/below right,C/above,
  D/below left,E/below right,F/above}{
    \fill[red] (\p) circle (2pt);
    \draw (\p) node[\placement] {$\p$};
  }
\end{tikzpicture}
```




```
\begin{tikzpicture}
  \draw[help lines] (0,0) grid (4,4);
  \coordinate (A) at (0,0);
  \coordinate (B) at (3,0);
  \coordinate (C) at (1,1);
```

```
  \coordinate [revolve/angle={C},
    revolve/scale=2,
    revolve={A,B}] (D);
  \draw[thick] (A) -- (B) (A) -- (D);
  \foreach \p/\placement in {A/left,B/right,
  C/above right,D/above}{
    \fill[red] (\p) circle (2pt);
    \draw (\p) node[\placement] {$\p$};
  }
\end{tikzpicture}
```



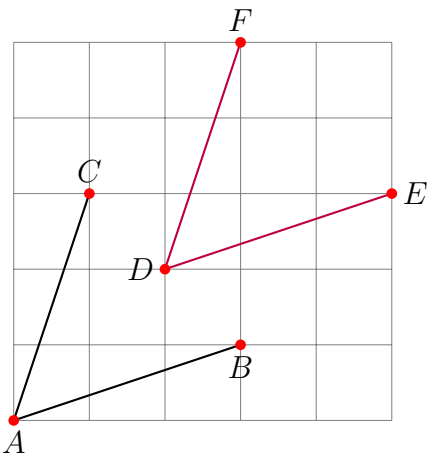
1.10 构造角 Angle

可以由 revolve 来构造一个角.

示例

```
\begin{tikzpicture}
  \draw[help lines] (0,0) grid (5,5);
  \coordinate (A) at (0,0);
  \coordinate (B) at (3,1);
  \coordinate (C) at (1,3);
  \coordinate (D) at (2,2);
  \coordinate (E) at (5,3);
```

```
    \coordinate [revolve/angle={A,B,C},
      revolve/scale=1,
      revolve={D,E}] (F);
    \draw[thick] (A) -- (B) (A) -- (C);
    \draw[thick, purple] (D) -- (E) (D) -- (F);
    \foreach \p/\placement in {A/below,B/below,C/above,
    D/left,E/right,F/above}{
      \fill[red] (\p) circle (2pt);
      \node[\placement] at (\p) {$\p$};
    }
  \end{tikzpicture}
```



1.11 角平分线 Angle Bisector

调用方式

```
angle bisector={A,B,C}
```

参数说明

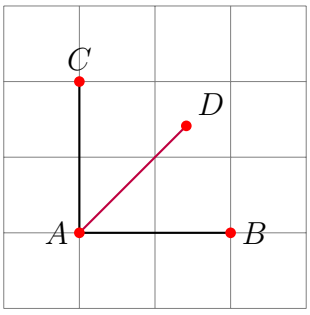
A,B,C 三点坐标, A 为顶点 (apex), B 为起点, C 为终点

返回 $\angle BAC$ 角平分线上的一点. 实际上, 该操作等价于:

```
revolve/angle={A,B,C}, revolve/scale=.5, revolve={A,B}
```

示例

```
\begin{tikzpicture}
  \draw[help lines] (0,0) grid (4,4);
  \coordinate (A) at (1,1);
  \coordinate (B) at (3,1);
  \coordinate (C) at (1,3);
  \coordinate [angle bisector={A,B,C}] (D);
  \draw[thick] (A) -- (B) (A) -- (C);
  \draw[thick, purple] (A) -- (D);
  \foreach \p/\placement in {A/left,B/right,C/above,
  D/above right}{
    \fill[red] (\p) circle (2pt);
    \draw (\p) node[\placement] {$\p$};
  }
\end{tikzpicture}
```



1.12 等边三角形 Equilateral Triangle

调用方式

```
equilateral={A,B}
```

参数说明

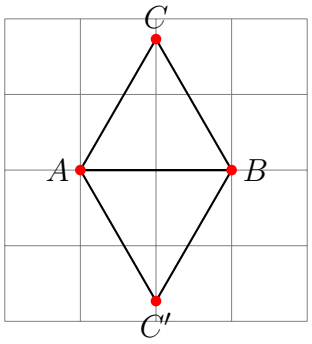
A,B 两点坐标

返回以 AB 为边长的等边三角形的第 3 点 (位于向量 AB 的左侧). 实际上, 该操作等价于:

```
revolve/angle=60, revolve={A,B}
```

示例

```
\begin{tikzpicture}
  \draw[help lines] (0,0) grid (4,4);
  \coordinate (A) at (1,2);
  \coordinate (B) at (3,2);
  \coordinate [equilateral={A,B}] (C);
  \coordinate [equilateral={B,A}] (C');
  \draw[thick] (A) -- (B) -- (C) -- cycle
               (A) -- (B) -- (C') -- cycle;
  \foreach \p/\placement in
   ↪  {A/left,B/right,C/above,C'/below}{
    \fill[red] (\p) circle (2pt);
    \draw (\p) node[\placement] {$\p$};
  }
\end{tikzpicture}
```



1.13 旋转 90° Erect

调用方式

```
erect={A,B}
```

参数说明

A,B 两点坐标

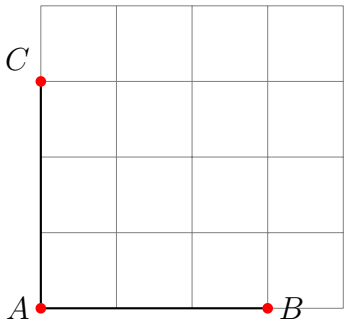
返回 B 绕 A 旋转 90° 的坐标. 实际上, 该操作等价于:

```
revolve/angle=90, revolve={A,B}
```

示例

```
\begin{tikzpicture}
  \draw[help lines] (0,0) grid (4,4);
  \coordinate (A) at (0,0);
  \coordinate (B) at (3,0);
  \coordinate [erect={A,B}] (C);
  \draw[thick] (A) -- (B) (A) -- (C);
```

```
    \foreach \p/\placement in {A/left,B/right,C/above
    ↪  left}{
      \fill[red] (\p) circle (2pt);
      \draw (\p) node[\placement] {$\p$};
    }
  \end{tikzpicture}
```



1.14 截取 Intercept

调用方式

```
intercept={A,B}
```

参数说明

A,B 两点坐标

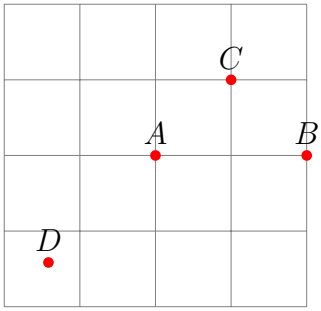
在直线 AB 截取指定长度线段, A 为新线段的起点, AB 是方向.

需要指定 intercept/length (default: 1cm) 和 intercept/scale(default: 1) 两个选项. 其中 intercept/length 有两种形式:

1. 直接指定长度: `intercept/length=2cm`
2. 指定线段长度: `intercept/length={P1,P2}`

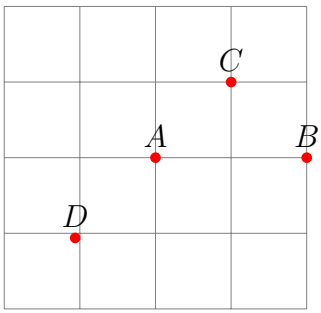
示例

```
\begin{tikzpicture}
  \draw[help lines] (-2,-2) grid (2,2);
  \coordinate (A) at (0,0);
  \coordinate (B) at (2,0);
  \coordinate (C) at (1,1);
  \coordinate[intercept/length={A,B},
    intercept/scale=-1, intercept={A,C}] (D);
  \foreach \p/\placement in
  ↪  {A/above,B/above,C/above,D/above}{
    \fill[red] (\p) circle (2pt);
    \draw (\p) node[\placement] {$\p$};
  }
\end{tikzpicture}
```




```
\begin{tikzpicture}
  \draw[help lines] (-2,-2) grid (2,2);
  \coordinate (A) at (0,0);
  \coordinate (B) at (2,0);
  \coordinate (C) at (1,1);
  \coordinate[intercept/length=1.5cm,
    intercept/scale=-1,
    intercept={A,C}] (D);
```

```
  \foreach \p/\placement in
  ↪  {A/above,B/above,C/above,D/above}{
    \fill[red] (\p) circle (2pt);
    \draw (\p) node[\placement] {$\p$};
  }
\end{tikzpicture}
```



1.15 直线与直线的交点 Line-Line Intersection

调用方式

```
intersect={A,B,C,D}
```

参数说明

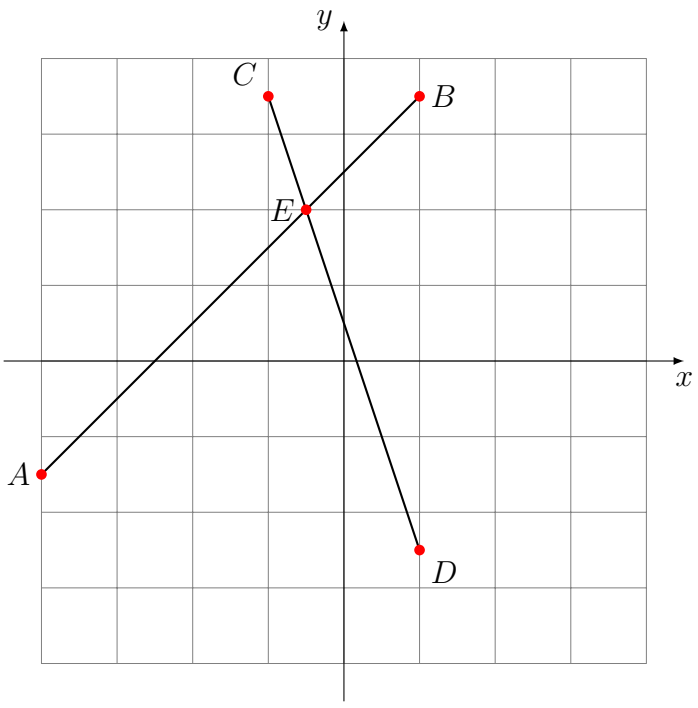
A,B,C,D 四点坐标

返回 AB 与 CD 的交点 (可以是延长线相交点).

示例

```
\begin{tikzpicture}
  \draw[help lines] (-4,-4) grid[step=1] (4,4);
  \draw[-latex] (-4.5,0) -- (4.5,0) node[below] {$x$};
  \draw[-latex] (0,-4.5) -- (0,4.5) node[left] {$y$};
```

```
    \coordinate (A) at (-4,-1.5);
    \coordinate (B) at (1,3.5);
    \coordinate (C) at (-1,3.5);
    \coordinate (D) at (1,-2.5);
    \coordinate [intersect={A,B,C,D}] (E);
    \draw[thick] (A) -- (B) (C) -- (D);
    \foreach \p/\placement in {A/left,B/right,
    C/above left,D/below right,E/left}{
      \fill[red] (\p) circle (2pt);
      \draw (\p) node[\placement] {$\p$};
    }
  \end{tikzpicture}
```



1.16 垂直平分线/中垂线 Perpendicular Bisector

调用方式

```
perpendicular bisector={A,B}
```

参数说明

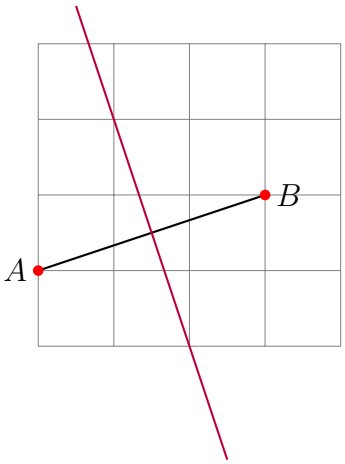
A,B 两点坐标

构造 AB 的中垂线, 默认起点为 $.5(A+B)+(B-A)\cdot\mathbf{i}$, 终点为 $.5(A+B)-(B-A)\cdot\mathbf{i}$. 可以对起始点进行调整, 见??.

示例

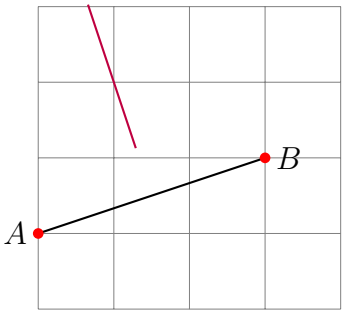
使用默认参数:

```
\begin{tikzpicture}
  \draw[help lines] (0,0) grid (4,4);
  \coordinate (A) at (0,1);
  \coordinate (B) at (3,2);
  \draw[thick] (A) -- (B);
  \draw[thick,purple,perpendicular bisector={A,B}];
  \foreach \p/\placement in {A/left,B/right}{
    \fill[red] (\p) circle (2pt);
    \draw (\p) node[\placement] {$\p$};
  }
\end{tikzpicture}
```



指定两端的长度:

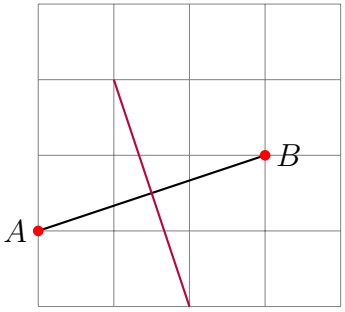
```
\begin{tikzpicture}
  \draw[help lines] (0,0) grid (4,4);
  \coordinate (A) at (0,1);
  \coordinate (B) at (3,2);
  \draw[thick] (A) -- (B);
  \draw[thick, purple,
    start modifier=.5cm, end modifier=2.5cm,
    perpendicular bisector={A,B}];
  \foreach \p/\placement in {A/left,B/right}{
    \fill[red] (\p) circle (2pt);
    \draw (\p) node[\placement] {$\p$};
  }
\end{tikzpicture}
```



指定系数:

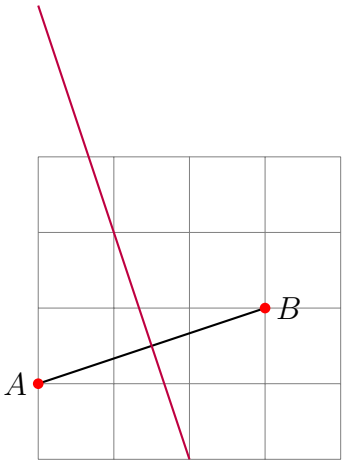
```
\begin{tikzpicture}
  \draw[help lines] (0,0) grid (4,4);
  \coordinate (A) at (0,1);
  \coordinate (B) at (3,2);
  \draw[thick] (A) -- (B);
  \draw[thick,purple,
```

```
    start modifier=.25,end modifier=.75,
    perpendicular bisector={A,B}];
  \foreach \p/\placement in {A/left,B/right}{
    \fill[red] (\p) circle (2pt);
    \draw (\p) node[\placement] {$\p$};
  }
\end{tikzpicture}
```



可以是负数, 这样就在相反方向:

```
\begin{tikzpicture}
  \draw[help lines] (0,0) grid (4,4);
  \coordinate (A) at (0,1);
  \coordinate (B) at (3,2);
  \draw[thick] (A) -- (B);
  \draw[thick,purple,
    start modifier=-.25,end modifier=0.75,
    perpendicular bisector={A,B}];
  \foreach \p/\placement in {A/left,B/right}{
    \fill[red] (\p) circle (2pt);
    \draw (\p) node[\placement] {$\p$};
  }
\end{tikzpicture}
```



1.17 垂线 Perpendicular Line

调用方式

```
perpendicular={A,B,C}
```

参数说明

A,B,C 三点坐标

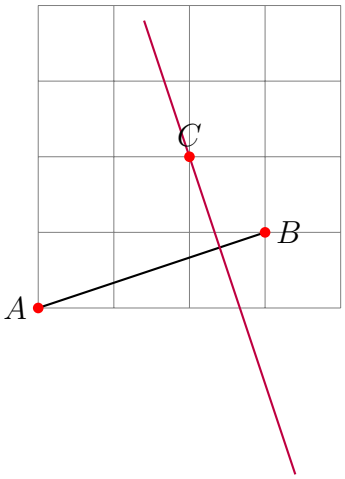
构造过 C 垂直于 AB 的直线 (设垂足为 D), 默认起点为 $D+(B-A)\cdot\mathbf{i}$, 终点为 $D-(B-A)\cdot\mathbf{i}$. 可以对起始点进行调整, 见??.

示例

过直线外一点的垂线:

```
\begin{tikzpicture}
  \draw[help lines] (0,0) grid (4,4);
  \coordinate (A) at (0,0);
  \coordinate (B) at (3,1);
  \coordinate (C) at (2,2);
  \draw[thick] (A) -- (B);
  \path[draw, thick, purple, perpendicular={A,B,C}];
```

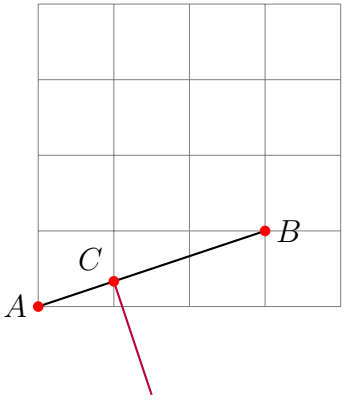
```
  \foreach \p/\placement in {A/left,B/right,C/above}{
    \fill[red] (\p) circle (2pt);
    \draw (\p) node[\placement] {$\p$};
  }
\end{tikzpicture}
```



过直线上一点的垂线:

```
\begin{tikzpicture}
  \draw[help lines] (0,0) grid (4,4);
  \coordinate (A) at (0,0);
  \coordinate (B) at (3,1);
  \coordinate (C) at ($(A)!1/3!(B)$);
  \draw[thick] (A) -- (B);
  \path[draw, thick, purple,
    start modifier=.5, end modifier=.75,
    perpendicular={A,B,C}];
  \foreach \p/\placement in {A/left,B/right,C/above
  ↪  left}{
    \fill[red] (\p) circle (2pt);
```

```
    \draw (\p) node[\placement] {$\p$};
  }
\end{tikzpicture}
```



1.18 平行线 Parallel Line

调用方式

```
parallel={A,B,C}
```

参数说明

过一点 C 作直线 AB 平行线, (如果 C 在 AB 上, 则重合).

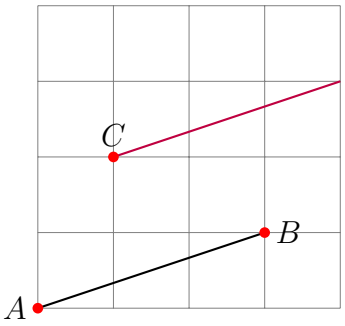
首先将点 C 按向量 AB 平移至 D. 可以对起始点进行调整, 见??.

示例

指定起始点距离 C 的位置, 方向是 CD, 负值代表相反方向:

```
\begin{tikzpicture}
  \draw[help lines] (0,0) grid (4,4);
  \coordinate (A) at (0,0);
  \coordinate (B) at (3,1);
```

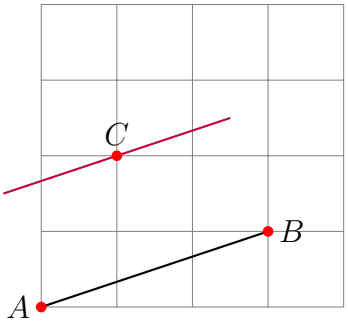
```
  \coordinate (C) at (1,2);
  \draw[thick] (A) -- (B);
  \path[draw, thick, purple, parallel={A,B,C}];
  \foreach \p/\placement in {A/left,B/right,C/above}{
    \fill[red] (\p) circle (2pt);
    \draw (\p) node[\placement] {$\p$};
  }
\end{tikzpicture}
```



指定系数:

```
\begin{tikzpicture}
  \draw[help lines] (0,0) grid (4,4);
  \coordinate (A) at (0,0);
  \coordinate (B) at (3,1);
  \coordinate (C) at (1,2);
  \draw[thick] (A) -- (B);
  \path[draw, thick, purple,
    start modifier=-.5, end modifier=.5,
    parallel={A,B,C}];
  \foreach \p/\placement in {A/left,B/right,C/above}{
    \fill[red] (\p) circle (2pt);
    \draw (\p) node[\placement] {$\p$};
```

```
  }
\end{tikzpicture}
```



1.19 延长线 Extend

调用方式

```
extend={A,B}
```

参数说明

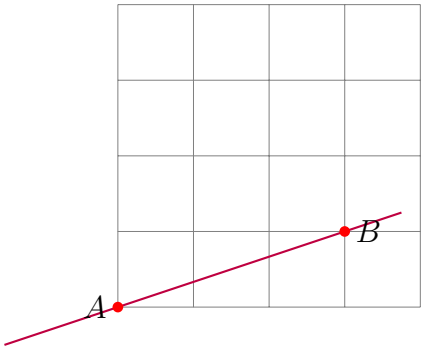
作线段 AB 延长线, 可以对起始点进行调整, 见??. 实际上, 该操作等价于:

```
parallel={A,B,A}
```

示例

```
\begin{tikzpicture}
  \draw[help lines] (0,0) grid (4,4);
  \coordinate (A) at (0,0);
  \coordinate (B) at (3,1);
  \path[draw, thick, purple,
```

```
    start modifier=-.5, end modifier=1.25,
    extend={A,B}];
  \foreach \p/\placement in {A/left,B/right}{
    \fill[red] (\p) circle (2pt);
    \draw (\p) node[\placement] {$\p$};
  }
\end{tikzpicture}
```



1.20 圆 Circle

调用方式

```
circle={O,A}
```

参数说明

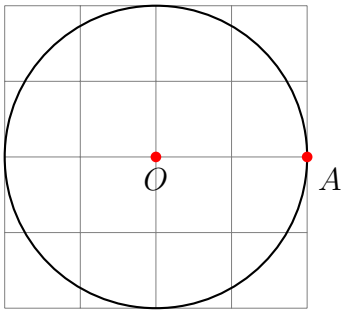
O 圆心

A 圆上一点

构造圆心为 O, 经过 A 的圆.

示例

```
\begin{tikzpicture}
  \draw[help lines] (-2,-2) grid (2,2);
  \coordinate (O) at (0,0);
  \coordinate (A) at +(0:2);  % 圆上一点，相对坐标
  \draw[thick,circle={O,A}];
  \foreach \p in {O,A}
    \fill[red] (\p) circle (2pt);
  \draw (O) node[below] {$O$};
  \draw (A) node[below right] {$A$};
\end{tikzpicture}
```



1.21 直线与圆的切点 Tangent Point

调用方式

```
circle-tangent = {O,A,P}
```

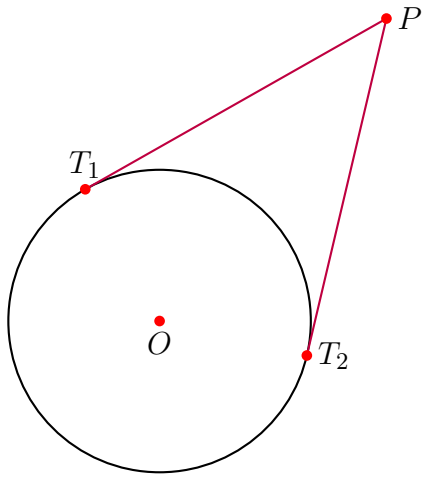
参数说明

O ：圆心坐标
A ：为圆上任意一点
P ：圆外一点坐标

过圆 (O 为圆心, A 为圆上任意一点) 外一点 P 作切线, 求得两个切点 (存储在 tp1 和 tp2 中, 其中 tp1 在向量 OP 的左边).

示例

```
\begin{tikzpicture}
  \coordinate (O) at (0,0);
  \coordinate (A) at +(0:2);  % 圆上一点，相对坐标
  \coordinate (P) at (3,4);
  \tikzset{circle-tangent={O,A,P}}
  \coordinate (T1) at (tp1);
  \coordinate (T2) at (tp2);
  \draw[thick,circle={O,A}];
  \draw[thick,purple] (P) -- (T1) (P) -- (T2);
  \foreach \p in {O,P,T1,T2}
    \fill[red] (\p) circle (2pt);
  \draw (O) node[below] {$O$};
  \draw (P) node[right] {$P$};
  \draw (T1) node[above] {$T_1$};
  \draw (T2) node[right] {$T_2$};
\end{tikzpicture}
```



调用方式

```
tangent point={O,A,P}
```

注 该调用方式已过时, 每次只能求出一个切点, 另一个切点需要用对称作出.

参数说明

O : 圆心坐标

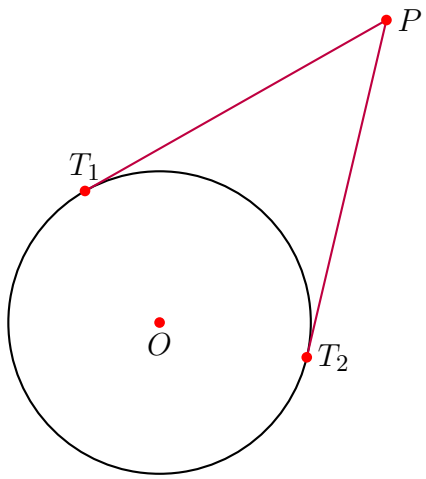
A : 为圆上任意一点

P : 圆外一点坐标

过圆 (O 为圆心, A 为圆上任意一点) 外一点 P 作切线, 求得一个切点 (在向量 OP 的左边), 另外一点可以通过对称 (`reflect={O,P,T}`) 求得.

示例

```
\begin{tikzpicture}
  \coordinate (O) at (0,0);
  \coordinate (A) at +(0:2);  % 圆上一点，相对坐标
  \coordinate (P) at (3,4);
  \coordinate[tangent point={O,A,P}] (T1);
  \coordinate[reflect={O,P,T1}] (T2);
  \draw[thick,circle={O,A}];
  \draw[thick,purple] (P) -- (T1) (P) -- (T2);
  \foreach \p in {O,P,T1,T2}
    \fill[red] (\p) circle (2pt);
  \draw (O) node[below] {$O$};
  \draw (P) node[right] {$P$};
  \draw (T1) node[above] {$T_1$};
  \draw (T2) node[right] {$T_2$};
\end{tikzpicture}
```



1.22 两圆的交点 Circle-Circle Intersection

调用方式

```
circle-circle={O1,A1,O2,A2}
```

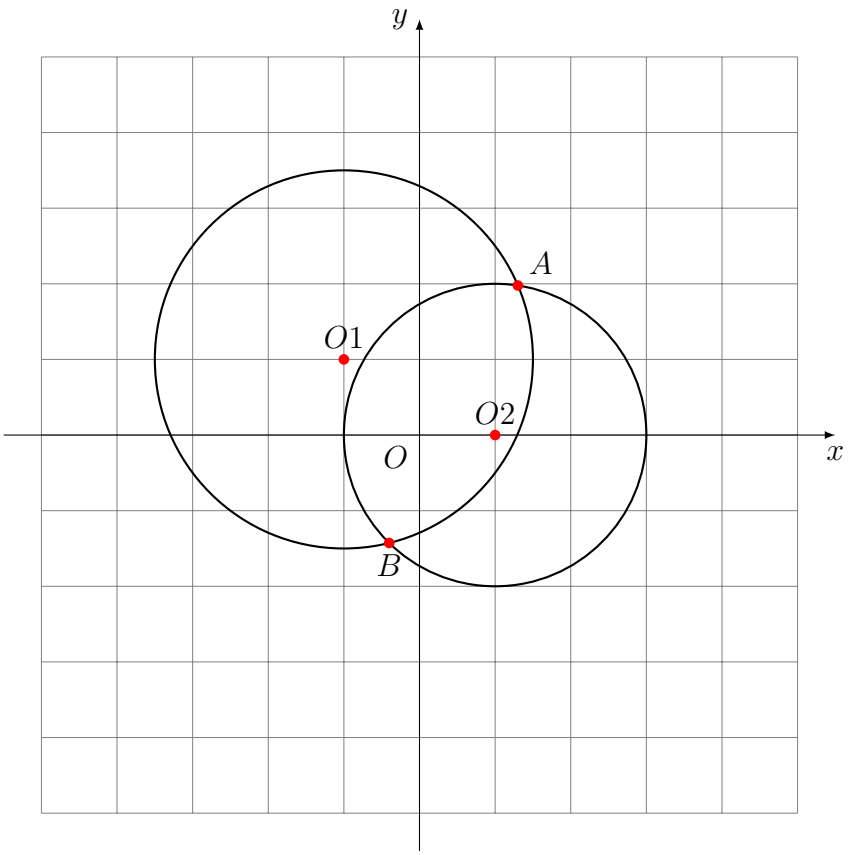
参数说明

求圆 1 (O_1 为圆心, A_1 为圆上任意一点) 和圆 2 (O_2 为圆心, A_2 为圆上任意一点) 交点坐标 (存储在 cc1 和 cc2 中, 以 Q_1Q_2 为正方向, cc1 在左侧, cc2 在右侧).

示例

```
\begin{tikzpicture}
  \tikzmath{
    \ra = 2.5;% radius a
    \rb = 2;% radius b
  }

  \axes(-5:5,-5:5);

  \coordinate (O1) at (-1,1);
```

```
    \coordinate (A1) at ($(O1)+(0:\ra)$);
    \coordinate (O2) at (1,0);
    \coordinate (A2) at ($(O2)+(0:\rb)$);

    \draw[thick] (O1) circle(\ra);
    \draw[thick] (O2) circle(\rb);

    \tikzset{circle-circle={O1,A1,O2,A2}}
    \coordinate (A) at (cc1);
    \coordinate (B) at (cc2);

    \foreach \p/\placement in {O1/above, O2/above, A/above
    ↪  right, B/below}{
      \fill[red] (\p) circle(2pt);
      \draw (\p) node[\placement] {$\p$};
    }
  \end{tikzpicture}
```



1.23 圆与直线的交点 Circle-Line Intersection

调用方式

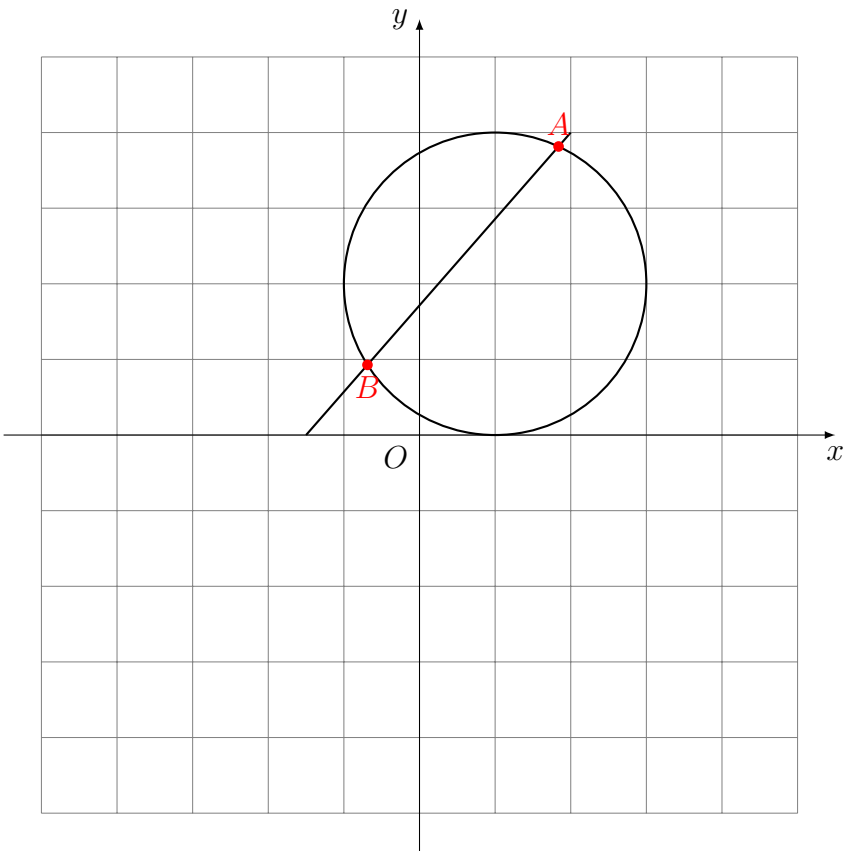
```
circle-line={O,A,P,Q}
```

参数说明

求圆 (O 为圆心, A 为圆上任意一点) 和直线 (P, Q 为直线上任意不重合的两点) 交点坐标 (存储在 cl1 和 cl2 中, 以 PQ 为正方向, cl1 在 cl2 的正方向上).

示例

```
\begin{tikzpicture}
  \tikzmath {
    \r = 2;
  }

  \axes(-5:5,-5:5);

  \coordinate (O) at (1,2);
  \coordinate (A) at ($(O)+(0:\r)$);
  \coordinate (P) at (-1.5,0);
  \coordinate (Q) at (2,4);
  \tikzset{circle-line={O,A,P,Q}}
  \draw[thick] (O) circle (\r);
  \draw[thick] (P) -- (Q);
  \fill[red] (cl1) circle(2pt) node[above] {$A$};
  \fill[red] (cl2) circle(2pt) node[below] {$B$};
\end{tikzpicture}
```



1.24 外位似中心 External Homothetic Center

调用方式

```
external center={O1,A1,O2,A2}
```

参数说明

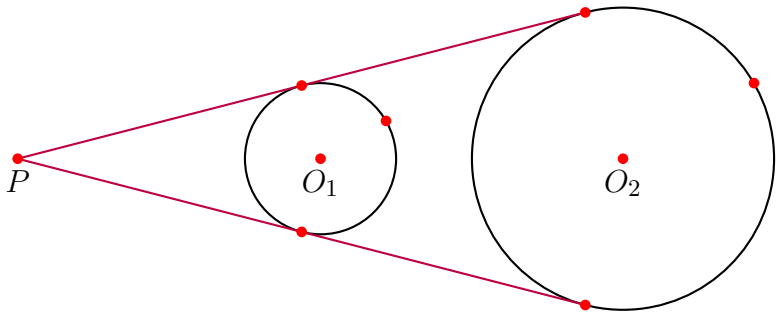
求圆 1 (O_1 为圆心, A_1 为圆上任意一点) 和圆 2 (O_2 为圆心, A_2 为圆上任意一点) 的外位似中心 (external homothetic center)[2].

示例

作外公切线: 先求位似中心, 可以求得两圆的外公切线.

```
\begin{tikzpicture}
  \tikzmath {
    \a = 30;
    \b = \a;
    \r1 = 1;
    \r2 = 2;
  }
  \coordinate (O1) at (0,0);
  \coordinate (A1) at ($(O1)+(\a:\r1)$);
  \coordinate (O2) at (4,0);
  \coordinate (A2) at ($(O2)+(\b:\r2)$);
  \coordinate[external center={O1,A1,O2,A2}] (P);

  \tikzset{circle-tangent={O1,A1,P},}
  \coordinate (B) at (tp1);
  \coordinate (C) at (tp2);

  \tikzset{circle-tangent={O2,A2,P},}
  \coordinate (D) at (tp1);
  \coordinate (E) at (tp2);

  \draw[thick,circle={O1,A1}];
  \draw[thick,circle={O2,A2}];

  \draw[thick,purple] (P) -- (D) (P) -- (E);
  \foreach \p in {A1,A2,B,C,D,E,O1,O2,P}
    \fill[red] (\p) circle (2pt);
  \draw (O1) node[below] {$O_1$};
  \draw (O2) node[below] {$O_2$};
  \draw (P) node[below] {$P$};
```

```
\end{tikzpicture}
```



1.25 内位似中心 Internal Homothetic Center

调用方式

```
internal center={O1,A1,O2,A2}
```

参数说明

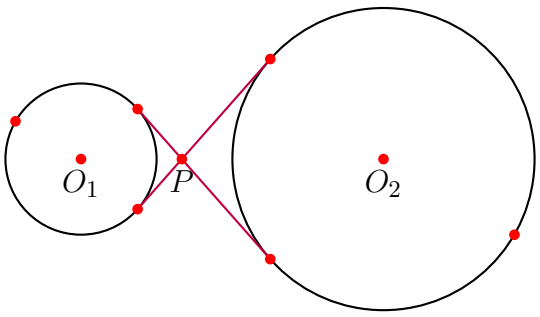
求圆 1 (O_1 为圆心, A_1 为圆上任意一点) 和圆 2 (O_2 为圆心, A_2 为圆上任意一点) 的内位似中心 (internal homothetic center)[2].

示例

作内公切线: 先求位似中心, 可以求得两圆的内公切线.

```
\begin{tikzpicture}
  \tikzmath {
    \a = 150;
    \b = \a - 180;
    \r1 = 1;
    \r2 = 2;
  }
  \coordinate (O1) at (0,0);
```

```
    \coordinate (A1) at ($(O1)+(\a:\r1)$);
    \coordinate (O2) at (4,0);
    \coordinate (A2) at ($(O2)+(\b:\r2)$);
    \coordinate[internal center={O1,A1,O2,A2}] (P);

    \tikzset{circle-tangent={O1,A1,P},}
    \coordinate (B) at (tp1);
    \coordinate (C) at (tp2);

    \tikzset{circle-tangent={O2,A2,P},}
    \coordinate (D) at (tp1);
    \coordinate (E) at (tp2);

    \draw[thick,circle={O1,A1}];
    \draw[thick,circle={O2,A2}];
    \draw[thick,purple] (P) -- (B) (P) -- (C) (P) -- (D)
    ↪  (P) -- (E);
    \foreach \p in {A1,A2,B,C,D,E,O1,O2,P}
      \fill[red] (\p) circle (2pt);
    \draw (O1) node[below] {$O_1$};
    \draw (O2) node[below] {$O_2$};
    \draw (P) node[below] {$P$};
\end{tikzpicture}
```



1.26 根轴 Radical Axis

调用方式

```
radical axis={O1,A1,O2,A2}
```

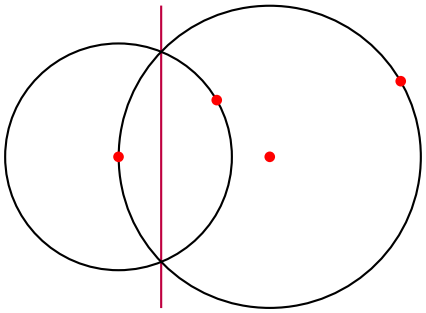
参数说明

构造两圆的根轴, 设与 O_1O_2 的交点为 P, 则默认起点为 $P+(O_2-O_1)\cdot\mathbf{i}$, 终点为 $P-(O_2-O_1)\cdot\mathbf{i}$. 可以对起始点进行调整, 见??.

示例

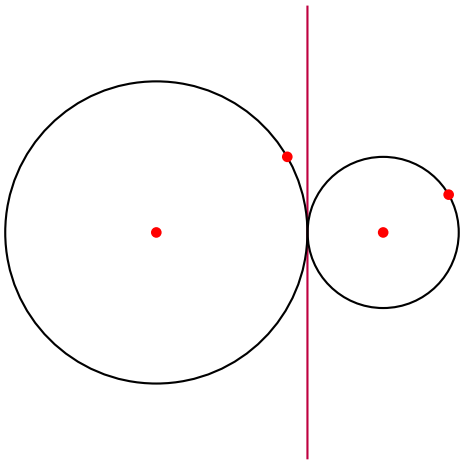
两圆相交:

```
\begin{tikzpicture}
  \tikzmath {
    \a = 30;
    \b = \a;
    \r1 = 1.5;
    \r2 = 2;
  }
  \coordinate (O1) at (0,0);
  \coordinate (A1) at ($(O1)+(\a:\r1)$);
  \coordinate (O2) at (2,0);
  \coordinate (A2) at ($(O2)+(\b:\r2)$);
  \draw[thick,purple,radical axis={O1,A1,O2,A2}];
  \draw[thick,circle={O1,A1}];
  \draw[thick,circle={O2,A2}];
  \foreach \p in {A1,A2,O1,O2}
    \fill[red] (\p) circle (2pt);
\end{tikzpicture}
```



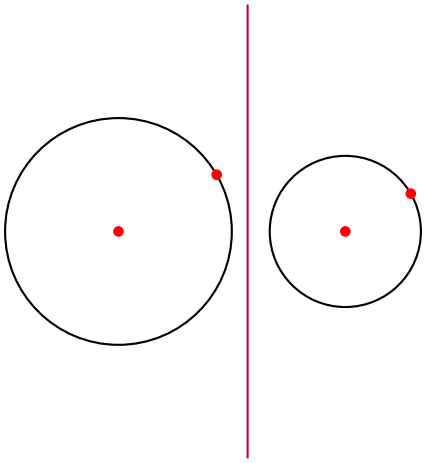
两圆外切:

```
\begin{tikzpicture}
  \tikzmath {
    \a = 30;
    \b = \a;
    \r1 = 2;
    \r2 = 1;
  }
  \coordinate (O1) at (0,0);
  \coordinate (A1) at ($(O1)+(\a:\r1)$);
  \coordinate (O2) at (3,0);
  \coordinate (A2) at ($(O2)+(\b:\r2)$);
  \draw[thick,purple,radical axis={O1,A1,O2,A2}];
  \draw[thick,circle={O1,A1}];
  \draw[thick,circle={O2,A2}];
  \foreach \p in {A1,A2,O1,O2}
    \fill[red] (\p) circle (2pt);
\end{tikzpicture}
```



两圆外离:

```
\begin{tikzpicture}
  \tikzmath {
    \a = 30;
    \b = \a;
    \r1 = 1.5;
    \r2 = 1;
  }
  \coordinate (O1) at (0,0);
  \coordinate (A1) at ($(O1)+(\a:\r1)$);
  \coordinate (O2) at (3,0);
  \coordinate (A2) at ($(O2)+(\b:\r2)$);
  \coordinate[radical axis={O1,A1,O2,A2}] (P);
  \draw[thick,purple,radical axis={O1,A1,O2,A2}];
  \draw[thick,circle={O1,A1}];
  \draw[thick,circle={O2,A2}];
  \foreach \p in {A1,A2,O1,O2}
    \fill[red] (\p) circle (2pt);
\end{tikzpicture}
```



1.27 Partway Modifiers and Distance Modifiers

`perpendicular bisector`,`perpendicular`,`parallel`,`radical axis` 等线段图形可以对起始点进行调整, 调整参数如下 [3]:

start modifier (default: 0), 长度或系数, 如: 1cm 或.75

end modifier (default: 1), 长度或系数, 如: 1cm 或.75

第 2 章　三角形的中心

2.1　重心 Centroid

调用方式

```
centroid={A,B,C}
```

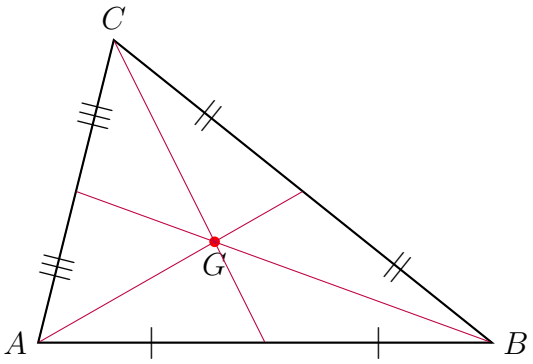
参数说明

A, B, C 三角形的顶点

示例

```
\begin{tikzpicture}
  \coordinate (A) at (-2,0);
  \coordinate (B) at (4,0);
  \coordinate (C) at (-1,4);
  \coordinate (D) at ($(B)!0.5!(C)$);
  \coordinate (E) at ($(C)!0.5!(A)$);
  \coordinate (F) at ($(A)!0.5!(B)$);
  \path[centroid={A,B,C}] coordinate (G);
  \fill (G) [red] circle (2pt);
  \draw (G) node[below] {$G$};
  \draw[thick] (A) -- (B) -- (C) -- cycle;
  \draw[purple] (A) -- (D) (B) -- (E) (C) -- (F);
  \draw (A) node[left] {$A$};
```

```
    \draw (B) node[right] {$B$};
    \draw (C) node[above] {$C$};
    \draw (A) -- (B) node[near start,sloped] {$|$}
    ↪  node[near end,sloped] {$|$};
    \draw (B) -- (C) node[near start,sloped] {$||$}
    ↪  node[near end,sloped] {$||$};
    \draw (C) -- (A) node[near start,sloped] {$|||$}
    ↪  node[near end,sloped] {$|||$};
  \end{tikzpicture}
```



2.2 垂心 Orthocenter

调用方式

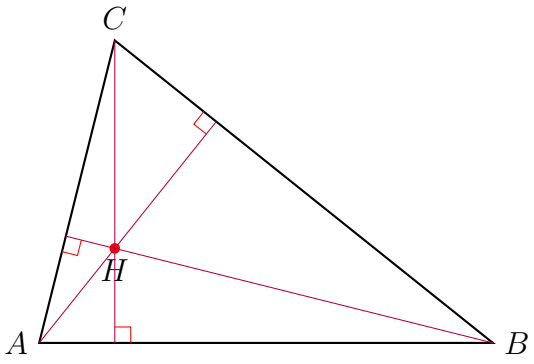
```
orthocenter={A,B,C}
```

参数说明

A, B, C 三角形的顶点

示例

```
\begin{tikzpicture}
  \coordinate (A) at (-2,0);
  \coordinate (B) at (4,0);
  \coordinate (C) at (-1,4);
  \path[orthocenter={A,B,C}] coordinate (H);
  \fill (H) [red] circle (2pt);
  \draw (H) node[below] {$H$};
  \draw[thick] (A) -- (B) -- (C) -- cycle;
  \coordinate (D) at ($(B)!(A)!(C)$);
  \coordinate (E) at ($(A)!(B)!(C)$);
  \coordinate (F) at ($(B)!(C)!(A)$);
  \draw[purple] (A) -- (D) (B) -- (E) (C) -- (F);
  \draw (A) node[left] {$A$};
  \draw (B) node[right] {$B$};
  \draw (C) node[above] {$C$};
  \pic [draw,red,angle radius=6pt] {right
  ↪ angle=H--D--C};
  \pic [draw,red,angle radius=6pt] {right
  ↪ angle=H--E--A};
  \pic [draw,red,angle radius=6pt] {right
  ↪ angle=H--F--B};
\end{tikzpicture}
```



2.3 外心 Circumcenter

调用方式

```
circumcenter={A,B,C}
```

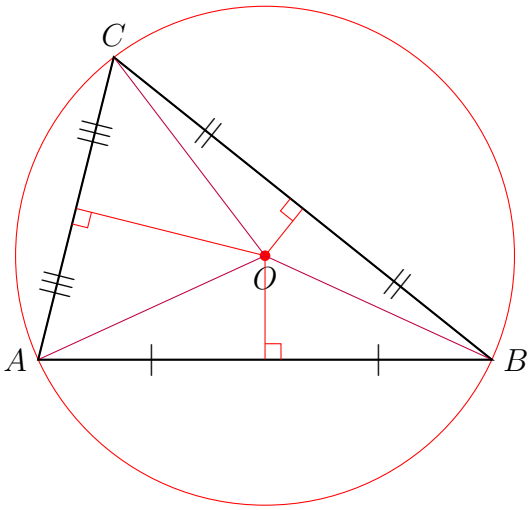
参数说明

A, B, C 三角形的顶点

示例

```
\begin{tikzpicture}
  \coordinate (A) at (-2,0);
  \coordinate (B) at (4,0);
  \coordinate (C) at (-1,4);
  \path[circumcenter={A,B,C}] coordinate (O);
  \fill (O) [red] circle (2pt);
  \draw (O) node[below] {$O$};
  \node[draw,red] at (O) [circle through=(A)]{};
  \draw[thick] (A) -- (B) -- (C) -- cycle;
  \draw[purple] (A) -- (O) (B) -- (O) (C) -- (O);
  \draw (A) node[left] {$A$};
  \draw (B) node[right] {$B$};
  \draw (C) node[above] {$C$};
  \coordinate (D) at ($(B)!(O)!(C)$);
  \coordinate (E) at ($(C)!(O)!(A)$);
  \coordinate (F) at ($(A)!(O)!(B)$);
  \draw[red] (O) -- (D) (O) -- (E) (O) -- (F);
  \draw (A) -- (B) node[near start,sloped] {$|$}
  ↪  node[near end,sloped] {$|$};
```

```
    \draw (B) -- (C) node[near start,sloped] {$||$}
    ↪ node[near end,sloped] {$||$};
    \draw (C) -- (A) node[near start,sloped] {$|||$}
    ↪ node[near end,sloped] {$|||$};
    \pic [draw,red,angle radius=6pt] {right
    ↪ angle=O--D--C};
    \pic [draw,red,angle radius=6pt] {right
    ↪ angle=O--E--A};
    \pic [draw,red,angle radius=6pt] {right
    ↪ angle=O--F--B};
\end{tikzpicture}
```



2.4 内心 Incenter

调用方式

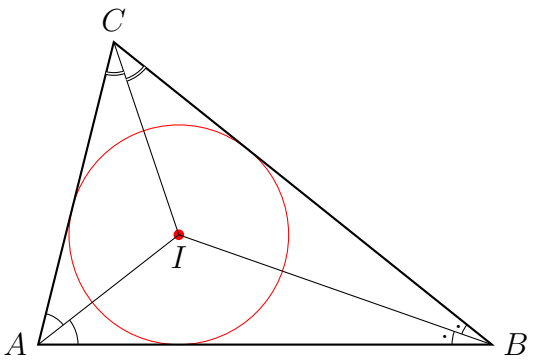
```
incenter={A,B,C}
```

参数说明

A, B, C 三角形的顶点

示例

```
\begin{tikzpicture}
  \coordinate (A) at (-2,0);
  \coordinate (B) at (4,0);
  \coordinate (C) at (-1,4);
  \path[incenter={A,B,C}] coordinate (I);
  \fill (I) [red] circle (2pt);
  \draw (I) node[below] {$I$};
  \node[draw,red] at (I) [circle
  ↪  through=($(B)!(I)!(C)$)]{};
  \draw[thick] (A) -- (B) -- (C) -- cycle;
  \draw (A) node[left] {$A$};
  \draw (B) node[right] {$B$};
  \draw (C) node[above] {$C$};
  \draw (A) -- (I) (B) -- (I) (C) -- (I);
  \pic [draw,angle radius=12pt] {angle=I--A--C};
  \pic [draw,angle radius=15pt] {angle=B--A--I};
  \pic [draw,double,angle radius=12pt] {angle=A--C--I};
  \pic [draw,double,angle radius=15pt] {angle=I--C--B};
  \pic [draw,pic text=.,angle radius=12pt,
    angle eccentricity=1.2] {angle=C--B--I};
  \pic [draw,pic text=.,angle radius=15pt,
    angle eccentricity=1.2] {angle=I--B--A};
\end{tikzpicture}
```



2.5 旁心 Excenter

调用方式

```
excenter={A,B,C}
```

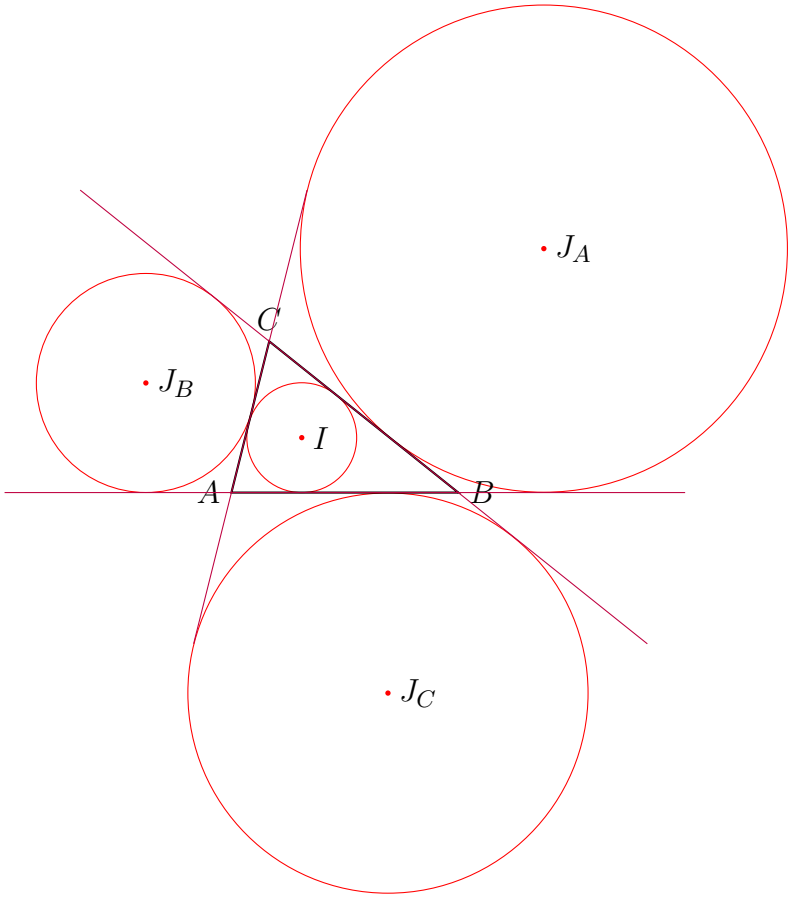
参数说明

A, B, C 三角形的顶点, 返回与 A 相对的旁心, 调换顶点顺序就可以得到 3 个旁心.

示例

```
\begin{tikzpicture}[scale=.5]
  \coordinate (A) at (-2,0);
  \coordinate (B) at (4,0);
  \coordinate (C) at (-1,4);
  \path[incenter={A,B,C}] coordinate (I);
  \path[excenter={A,B,C}] coordinate (JA);
  \path[excenter={B,A,C}] coordinate (JB);
  \path[excenter={C,A,B}] coordinate (JC);
  \foreach \point in {I,JA,JB,JC}
    \fill (\point) [red] circle (2pt);
```

```
    \node[draw,red] at (I) [circle
    ↪ through=($(B)!(I)!(C)$)]{};
    \node[draw,red] at (JA) [circle
    ↪ through=($(B)!(JA)!(C)$)]{};
    \node[draw,red] at (JB) [circle
    ↪ through=($(B)!(JB)!(C)$)]{};
    \node[draw,red] at (JC) [circle
    ↪ through=($(B)!(JC)!(C)$)]{};
    \draw[thick] (A) -- (B) -- (C) -- cycle;
    \draw (A) node[left] {$A$};
    \draw (B) node[right] {$B$};
    \draw (C) node[above] {$C$};
    \draw[purple] ($(A)!-1!(B)$) -- ($(A)!2!(B)$);
    \draw[purple] ($(B)!-1!(C)$) -- ($(B)!2!(C)$);
    \draw[purple] ($(C)!-1!(A)$) -- ($(C)!2!(A)$);
    \draw (I) node[right] {$I$};
    \draw (JA) node[right] {$J_A$};
    \draw (JB) node[right] {$J_B$};
    \draw (JC) node[right] {$J_C$};
\end{tikzpicture}
```



2.6 九点圆圆心 Nine-Point Center

调用方式

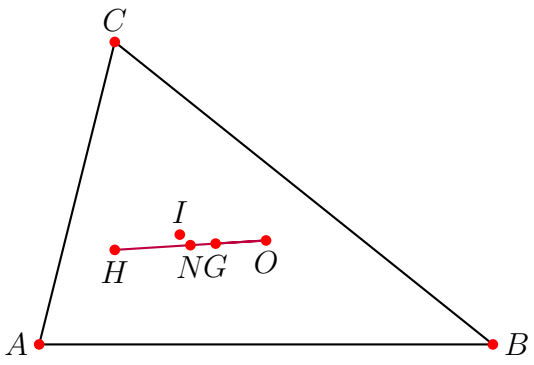
```
nine-point center={A,B,C}
```

参数说明

A, B, C 三角形的顶点

示例

```
\begin{tikzpicture}
  \coordinate (A) at (-2,0);
  \coordinate (B) at (4,0);
  \coordinate (C) at (-1,4);
  \path[orthocenter={A,B,C}] coordinate (H);
  \path[circumcenter={A,B,C}] coordinate (O);
  \path[centroid={A,B,C}] coordinate (G);
  \path[incenter={A,B,C}] coordinate (I);
  \path[nine-point center={A,B,C}] coordinate (N);
  \draw[thick] (A) -- (B) -- (C) -- cycle;
  \draw[thick,purple] (H) -- (O) -- (G);
  \foreach \p/\placement in {A/left,B/right,C/above,
  H/below,O/below,G/below,I/above,N/below}{
    \fill (\p) [red] circle (2pt);
    \draw (\p) node[\placement] {$\p$};
  }
\end{tikzpicture}
```



第 3 章　圆锥曲线 Conics

尽管 tikz 内置的 `ellipse` 和 `parabola` 绘制椭圆和抛物线, 这里定义了 `\ellipse` 和 `\parabola`.

3.1　椭圆 Ellipse

3.1.1　已知半长轴长 a 和半短轴长 b 绘图

调用方式

```
\ellipse [options] (a,b)
```

参数说明

a, b 半长轴长 (semi-major axis) 和半短轴长 (semi-minor axis), 默认单位为 cm, 可指定单位, 如 `(4cm, 3cm)`

返回中心为原点的椭圆曲线: $\dfrac{x^2}{a^2}+\dfrac{y^2}{b^2}=1$.

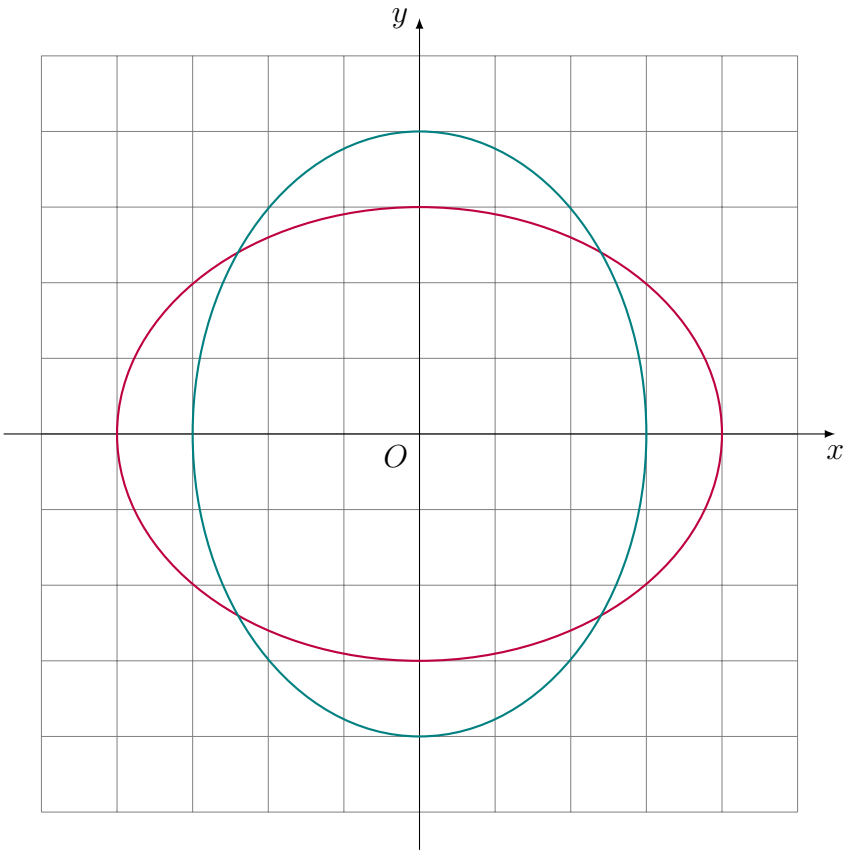
注 当指定椭圆曲线 (ellipse) 的 `domain` (default: `domain=-180:180`) 时, `domain` 是下列参数方程中 t 的取值范围:

$$\begin{cases} x = a\cos t, \\ y = b\sin t \end{cases}$$

示例

使用 tikz 内置曲线:

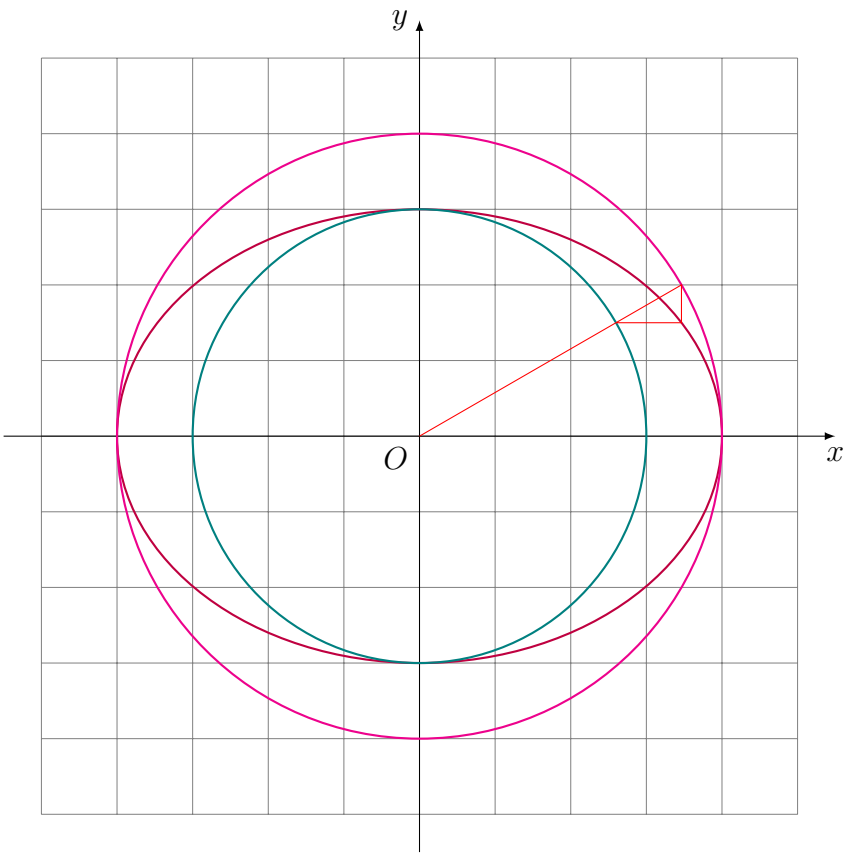
```
\begin{tikzpicture}
  \axes (-5:5,-5:5);
  \draw[thick,purple] (0,0) ellipse (4cm and 3cm);
  \draw[thick,teal] (0,0) ellipse [x radius=3cm,y
  ↪ radius=4cm];
\end{tikzpicture}
```



使用 \ellipse 命令:

```
\begin{tikzpicture}
  \axes (-5:5,-5:5);
  \ellipse[draw,thick,purple] (4,3);
```

```
    \ellipse[draw,thick,teal] (3,3);
    \ellipse[draw,thick,magenta] (4,4);

    \coordinate (O) at (0,0);
    \coordinate (A) at ({4*cos(30)},{4*sin(30)});
    \coordinate (B) at ({3*cos(30)},{3*sin(30)});
    \coordinate (C) at ({4*cos(30)},{3*sin(30)});
    \draw[red] (O) -- (A) -- (C) -- (B);
  \end{tikzpicture}
```



3.1.2 由两焦点和一点确定椭圆

调用方式

```
`ellipse/define = {F1,F2,P}`: define an ellipse with
↪ two foci and a point.
`ellipse`: creates the ellipse path.
`ellipse/directrix/scale=k`: set the scale of
↪ directrices.
`ellipse/directrix`: create the directrices.
`ellipse/axis/scale=k`: set the scale of axes.
`ellipse/axis`: create the axes.
`ellipse/a`, `ellipse/b`, `ellipse/c`, `ellipse/e`:
↪ ellipse semimajor-axis, semiminor-axis, linear
↪ eccentricity, eccentricity.
```

注意:

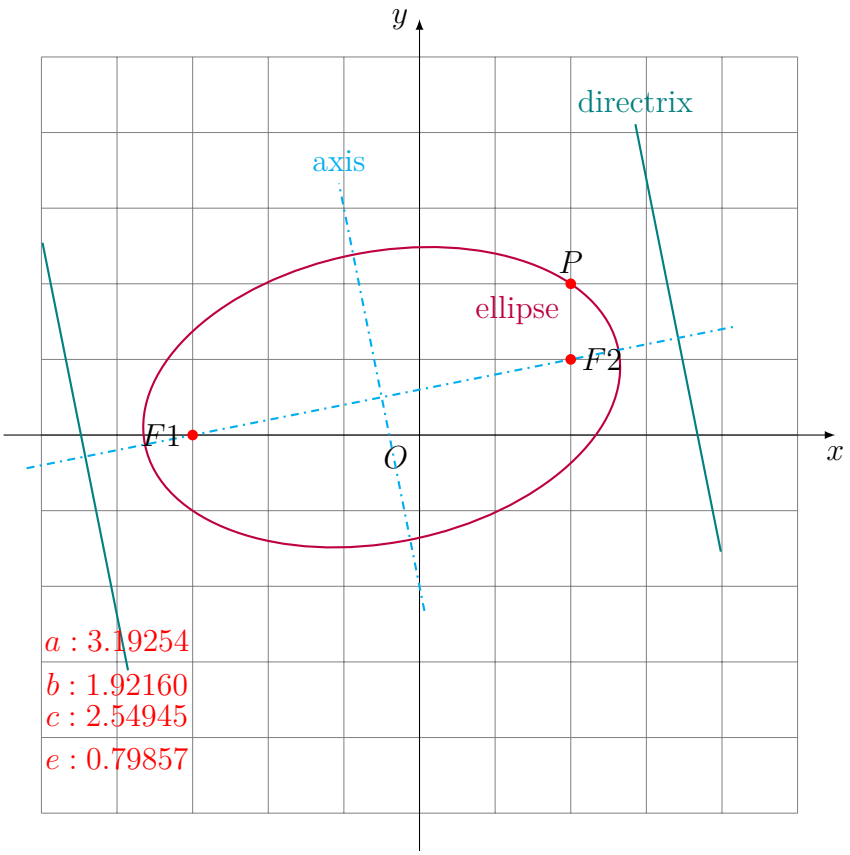
```
ellipse/define = {F1,F2,P}
```

必须单独使用 tikzset 定义才能访问 /tikz/ellipse 下面的变量, 这和作用域有关.

示例

```
\begin{tikzpicture}
  \axes(-5:5, -5:5);

  \coordinate (F1) at (-3,0);
  \coordinate (F2) at (2,1);
  \coordinate (P) at (2,2);

  \tikzset{
    ellipse/define ={F1,F2,P},
  }
```

```
    \draw [purple, thick, ellipse] node at (P) [below
    ↪ left] {ellipse};
    \draw [teal, thick,ellipse/directrix/scale=1.
    ↪ 5,ellipse/directrix] node [above] {directrix};
    \draw [cyan, thick, dashdotted,
    ↪ ellipse/axis/scale=1.5,ellipse/axis] node [above]
    ↪ {axis};

    % ellipse parameters
    \draw[red] (-4,-3) node[above]
    ↪ {$a:{\pgfkeysvalueof{/tikz/ellipse/a}}$};
    \draw[red] (-4,-3) node[below]
    ↪ {$b:{\pgfkeysvalueof{/tikz/ellipse/b}}$};
    \draw[red] (-4,-4) node[above]
    ↪ {$c:{\pgfkeysvalueof{/tikz/ellipse/c}}$};
    \draw[red] (-4,-4) node[below]
    ↪ {$e:{\pgfkeysvalueof{/tikz/ellipse/e}}$};

    \foreach \p/\placement in {F1/left,F2/right,P/above}{
      \fill[red] (\p) circle (2pt);
      \draw (\p) node[\placement] {$\p$};
    }

  \end{tikzpicture}
```



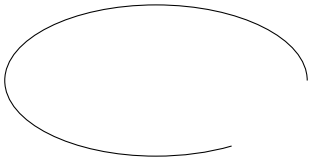
3.2 椭圆弧 Arc

基本用法

在 TikZ 中绘制圆弧可以通过 arc 操作来实现, 将椭圆的一部分添加到当前路径中.

```
\begin{tikzpicture}
  % Arc operation
  \draw (2,0) arc
    [
      start angle=0,
      end angle=300,
      x radius=2cm,
```

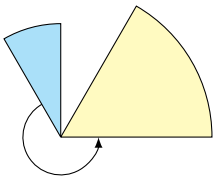
```
      y radius =1cm
    ];
\end{tikzpicture}
```



圆弧从点 (2,0) 开始, 以 0 度（由起始角度指定）结束, 以 300 度（由结束角度指定）结束.

为了绘制圆的一部分, 我们使用与前一个相同的语法, 并且不提供参数 x radius 和 y radius, 而是仅提供参数 radius.

```
\begin{tikzpicture}

  \draw[fill=yellow!30] (0,0) -- (2,0) arc[start
  ↪  angle=0, end angle=60,radius=2cm] -- (0,0);

  \draw[fill=cyan!30] (0,0) -- (0,1.5) arc [start
  ↪  angle=90, delta angle=30, radius=1.5cm] -- (0,0);

  \draw[-latex] (120:0.5) arc (120:360:0.5) ;

\end{tikzpicture}
```



- 黄色扇区是通过 arc 操作绘制的, 为了得到圆的一部分, 我们只定义 radius 而不是 x radius 和 y radius 参数.

- 蓝色扇区的绘制方式与黄色扇区相同, 但在本例中我们指定了 delta angle 而不是 end angle, 后者等于 start angle + delta angle.

简短语法

上图中的弯曲箭头是使用 arc 操作的较短语法绘制的, 对应于圆的一部分:

arc(start angle:end angle: radius)

对于椭圆, 我们使用以下语法:

arc(start angle:end angle: x radius and y radius)

然而, 这种语法不直观且难以阅读, 因此一般情况下应首选普通语法 (见 PGF 手册).

3.3 抛物线 Parabola

3.3.1 已知方程系数 a, b 和 c 绘图

调用方式

```
\parabola [options] (a,b,c)
```

参数说明

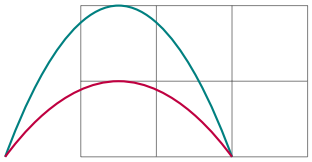
a,b,c 二次函数的系数

返回抛物线: $y = ax^2 + bx + c$.

示例

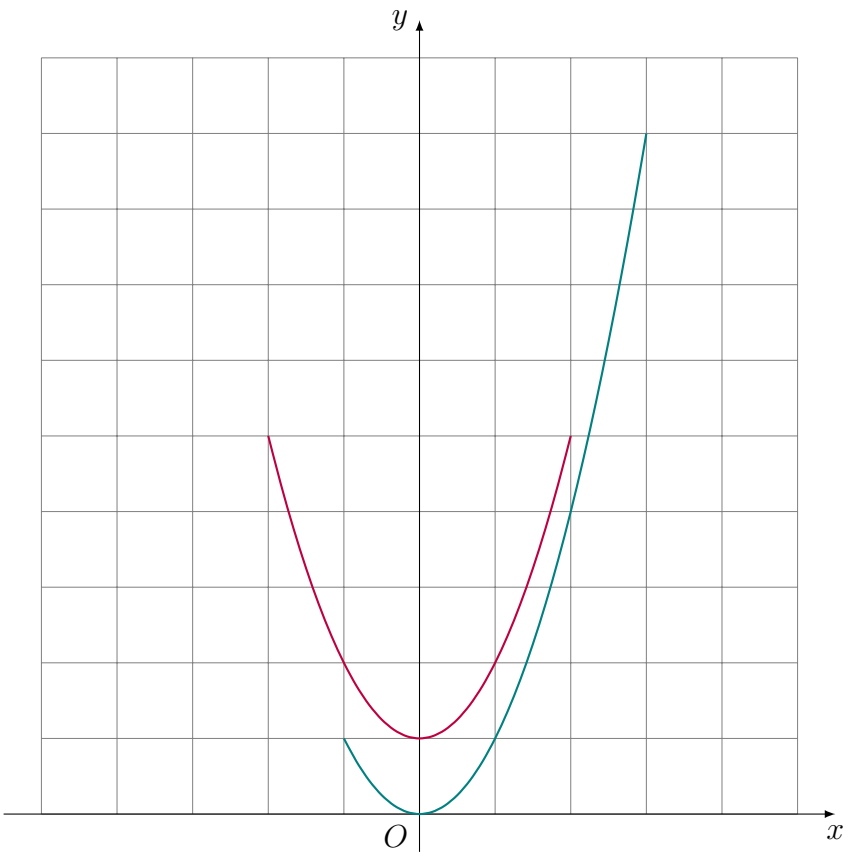
使用 tikz 内置曲线:

```
\begin{tikzpicture}
  \draw[help lines] (0,0) grid (3,2);
  \draw[thick,teal] (-1,0) parabola[bend pos=0.5] bend
  ↪ +(0,2) +(3,0);
  \draw[thick,purple] (-1,0) parabola[parabola
  ↪ height=1cm] +(3,0);
\end{tikzpicture}
```



使用 \parabola 命令:

```
\begin{tikzpicture}
  \axes (-5:5,0:10);
  \parabola[draw,thick,teal,domain=-1:3] (1,0,0);
  \parabola[draw,thick,purple] (1,0,1);
\end{tikzpicture}
```



3.3.2 由焦点和顶点确定抛物线

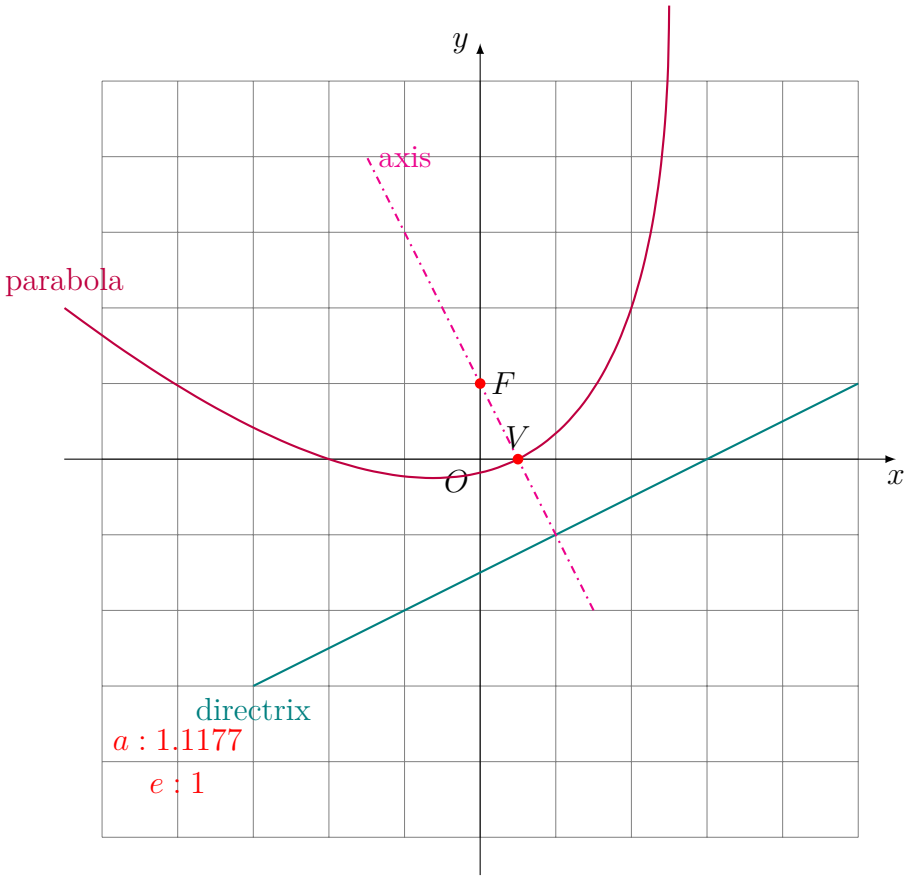
调用方式

```
`parabola/define = {F,P}`: define a parabola with the
↪ focus and vertex.
`parabola/domain=t1:t2`: set the domain for parametric
↪ equation: x = a*t^2, y = 2a*t, where `a` is the
↪ distance from the focus to the vertex.
`parabola`: creates the parabola path.
`parabola/directrix/scale=k`: set the scale of
↪ directrices.
`parabola/directrix`: create the directrix.
`parabola/axis/scale=k`: set the scale of axis.
`parabola/axis`: create the axis.
`parabola/a`, `parabola/e`: 1/4 parabola latus rectum,
↪ eccentricity (1.0).
```

示例

```
\begin{tikzpicture}
  \axes(-5:5, -5:5);

  \coordinate (F) at (0,1);% focus
  \coordinate (V) at (0.5,0);% vertex

  \tikzset{
    % parabola/doamin=-1.25:1.25,
    parabola/define={F,V},
  }
```

```
    \draw [purple, thick, parabola] node [above]
    ↪ {parabola};
    \draw [teal, thick,
    % parabola/directrix/scale=1.5,
    parabola/directrix] node [below] {directrix};
    \draw [magenta, thick, dashdotted,
    % parabola/axis/scale=2.5,
    parabola/axis] node [right] {axis};

    \draw[red] (-4,-4) node[above]
    ↪ {$a:{\pgfkeysvalueof{/tikz/parabola/a}}$};
    \draw[red] (-4,-4) node[below]
    ↪ {$e:{\pgfkeysvalueof{/tikz/parabola/e}}$};

    \foreach \pp/\placement in {F/right,V/above}{
      \fill[red] (\pp) circle (2pt);
      \draw (\pp) node[\placement] {$\pp$};
    }

  \end{tikzpicture}
```



3.4 双曲线 Hyperbola 与渐近线 Asymptote

3.4.1 已知半实轴长 a 和半虚轴长 b 绘图

调用方式

```
\hyperbola [options] (a,b);
```

或

```
\asymptote [options] (a,b);
```

参数说明

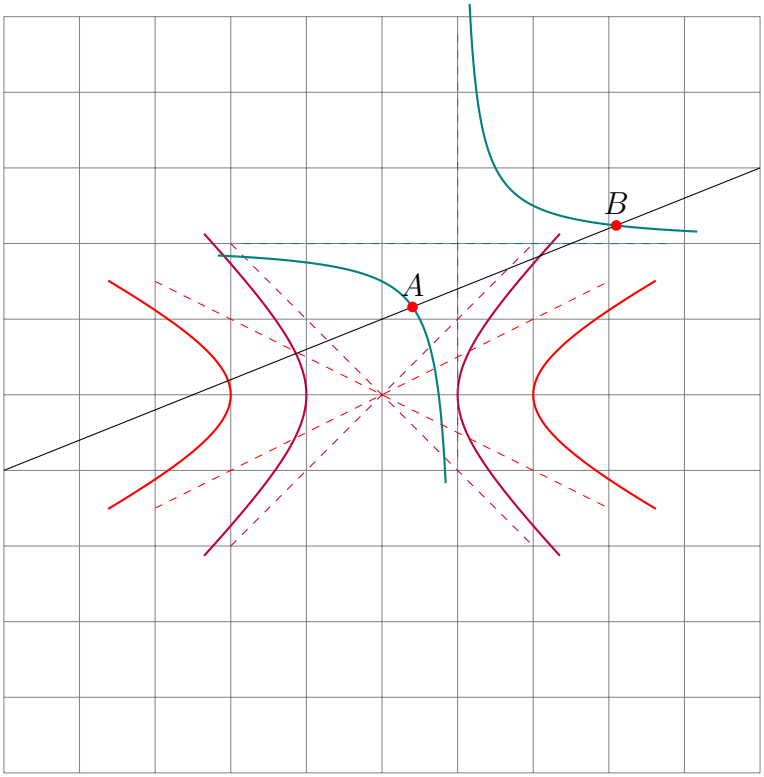
a, b 半实轴长 (semi-major axis) 和半虚轴长 (semi-minor axis), 默认单位为 cm, 可指定单位, 如 `(4cm, 3cm)`

分别返回中心在原点的双曲线: $\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1$ 和渐近线: $y = \pm\frac{b}{a}x$.

示例

```
\begin{tikzpicture}
  \tikzmath{
    \a = 1;
    \b = 1;
  }
  \draw[help lines] (-5,-5) grid (5,5);

  \hyperbola [draw,thick,purple,name path=c1] (\a,\b);
  \asymptote [draw,dashed,purple] (\a,\b);

  \hyperbola [transform={45:(1,2)},draw,thick,teal,name
  ↪  path=c2] (1,1);
  \asymptote [transform={45:(1,2)},draw,dashed,teal]
  ↪  (1,1);

  \hyperbola [draw,thick,red,domain=-1.2:1.2,name
  ↪  path=c3] (2,1);
  \asymptote [draw,dashed,red,domain=-3:3] (2,1);

  \draw[name path=l] (-5,-1) -- (5,3);
  \path[name intersections={of=c2 and l, by={A,B}, sort
  ↪  by=l}];
  \foreach \p/\placement in {A/above, B/above}{
    \fill[red] (\p) circle (2pt);
```

```
    \draw (\p) node[\placement] {$\p$};
  }
\end{tikzpicture}
```



注 当指定绘制双曲线 (hyperbola) 的 `domain` (default: `domain=-1.5:1.5`) 时, `domain` 是下列双曲线参数方程中 t 的取值范围:

$$\begin{cases} x = \pm a\cosh t, \\ y = b\sinh t \end{cases}$$

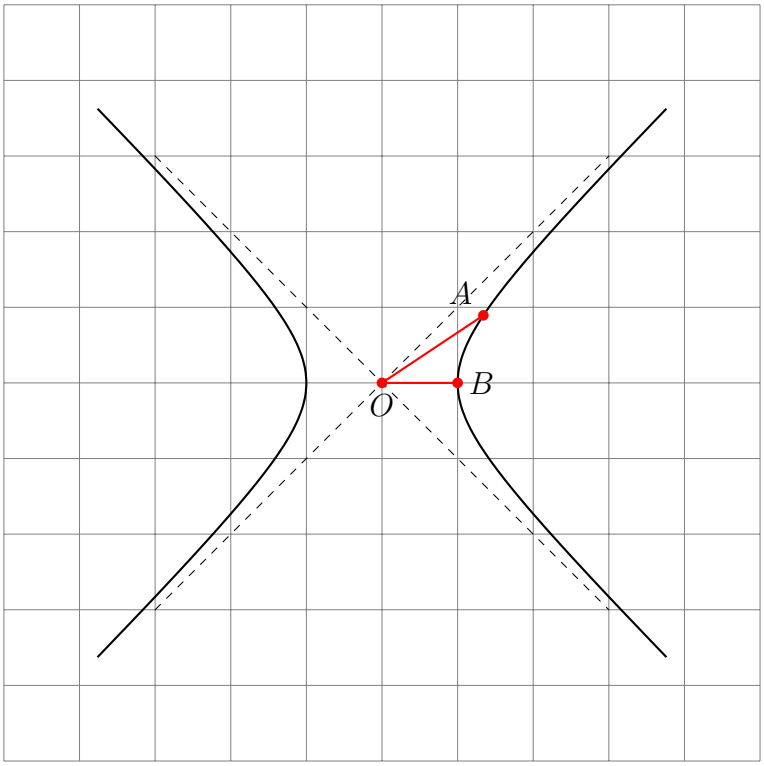
t 的几何意义: 射线出原点交单位双曲线 $x^2-y^2=1$ 于 $(x=\cosh t, y=\sinh t)$, t 是射线, 双曲线和 x 轴围成的面积的二倍. 对于双曲线上位于 x 轴下方的点, 这个面积被认为是负值.

当指定绘制渐进线 (asymptote) 的 `domain` (default: `domain=-2:2`) 时, `domain` 是下列直线方程中 x 的取值范围:

$$y = \pm\frac{b}{a}x$$

示例

```
\begin{tikzpicture}
  \draw[help lines] (-5,-5) grid (5,5);

  \coordinate (O) at (0,0);
  \coordinate (A') at (3,2);
  \coordinate (B) at (1,0);

  \path[name path=ray] (O) -- (A');

  \hyperbola[draw,thick,domain=-2:2,name
  ↪ path=hyperbola] (1,1);
  \asymptote[draw,dashed,domain=-3:3] (1,1);

  \path[name intersections={of=ray and hyperbola,
  ↪ by=A}];

  \draw[thick,red] (A) -- (O) -- (B);

  \foreach \p/\placement in {O/below, A/above left,
  ↪ B/right}{
    \fill[red] (\p) circle (2pt);
    \draw (\p) node[\placement] {$\p$};
  }
\end{tikzpicture}
```



也可使用 pgfplots 底层函数来绘制双曲线:

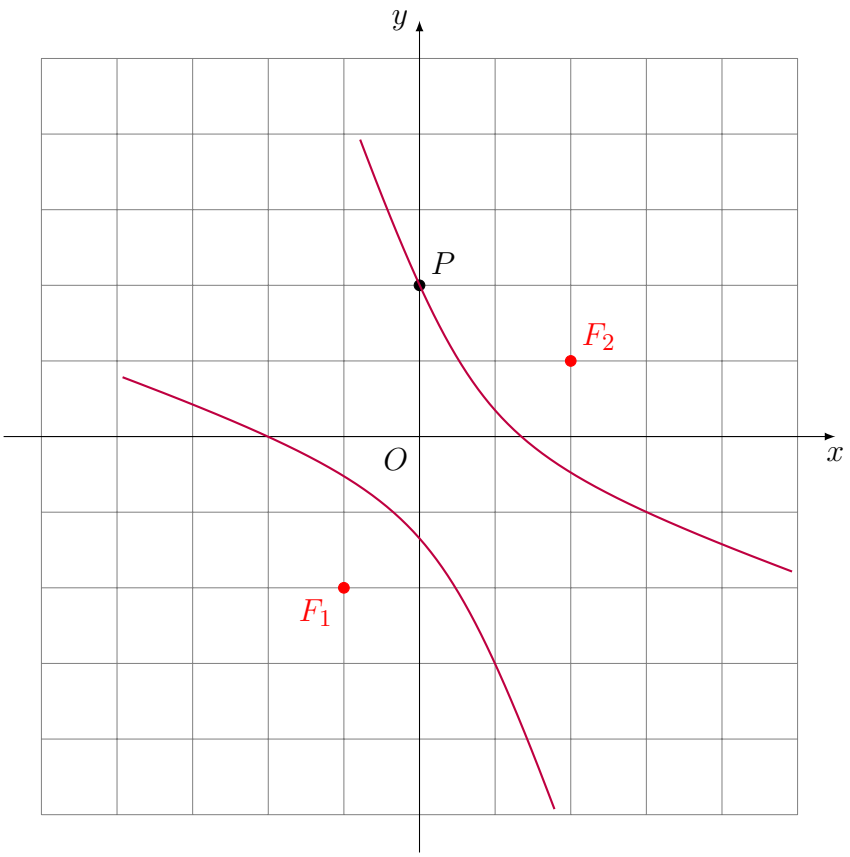
示例

```
\begin{tikzpicture}
  \axes (-5:5,-5:5);

  % 定义焦点 F1 和 F2, 以及点 P
  \coordinate (F1) at (-1, -2); % Left focus
  \coordinate (F2) at (2, 1); % Right focus
  \coordinate (P) at (0, 2);

  % 提取各点的坐标分量 x-coordinate and y-coordinate
  % 注意: 根据 LaTeX 的规则, 宏不能含有数字, 提取坐标的 x
  ↪ 和 y 分量的宏不要含有数字
  \newdimen\xFL
```

```
\newdimen\yFL
\newdimen\xFR
\newdimen\yFR
\newdimen\xP
\newdimen\yP
\pgfextractx{\xFL}{\pgfpointanchor{F1}{center}}
\pgfextracty{\yFL}{\pgfpointanchor{F1}{center}}
\pgfextractx{\xFR}{\pgfpointanchor{F2}{center}}
\pgfextracty{\yFR}{\pgfpointanchor{F2}{center}}
\pgfextractx{\xP}{\pgfpointanchor{P}{center}}
\pgfextracty{\yP}{\pgfpointanchor{P}{center}}

% 计算 2a = |PF1 - PF2|
% 注意: \x{} \y{} 只能在 let 中
%\pgfmathsetmacro{\PFLeft}{veclen(\x{P}-\x{F1},
↪  \y{P}-\y{F1})} % 计算 PF1
%\pgfmathsetmacro{\PFRight}{veclen(\x{P}-\x{F2},
↪  \y{P}-\y{F2})} % 计算 PF2
\pgfmathsetmacro{\PFLeft}{veclen(\xP-\xFL, \yP-\yFL)}
↪  % 计算 PF1
\pgfmathsetmacro{\PFRight}{veclen(\xP-\xFR,
↪  \yP-\yFR)} % 计算 PF2
\pgfmathsetmacro{\a}{abs(\PFLeft - \PFRight)/2cm} % 计
↪  算 a 并转换单位

% 计算焦距 c
%\pgfmathsetmacro{\c}{veclen(\x{F1}-\x{F2},
↪  \y{F1}-\y{F2}) / 2} % c = |F1F2| / 2
\pgfmathsetmacro{\c}{veclen(\xFL-\xFR, \yFL-\yFR) /
↪  2cm} % c = |F1F2| / 2
```

```
% 计算 b
\pgfmathsetmacro{\b}{sqrt(\c^2 - \a^2)} % b = sqrt(c^2
↪  - a^2)

% 计算双曲线的中心 (h, k)
\pgfmathsetmacro{\h}{(\xFL + \xFR) / 2cm} % h
\pgfmathsetmacro{\k}{(\yFL + \yFR) / 2cm} % k

% 计算旋转角度 theta
%\pgfmathsetmacro{\theta}{atan2(\y{F2}-\y{F1},
↪  \x{F2}-\x{F1})} % theta
\pgfmathsetmacro{\theta}{atan2(\yFR-\yFL, \xFR-\xFL)}
↪  % theta

% 绘制焦点
\filldraw[red] (F1) circle (2pt) node[below
↪  left]{$F_1$};
\filldraw[red] (F2) circle (2pt) node[above
↪  right]{$F_2$};

% 绘制点 P
\filldraw[black] (P) circle (2pt) node[above
↪  right]{$P$};

% 绘制双曲线
\draw[domain=-1.5:1.5, smooth, variable=\t, purple,
↪  thick, rotate around={\theta:({\h},{\k})}]
  plot ({\a*cosh(\t) + \h}, {\b*sinh(\t) + \k});
```

```
    \draw[domain=-1.5:1.5, smooth, variable=\t, purple,
    ↪  thick, rotate around={\theta:({\h},{\k})}]
      plot ({-\a*cosh(\t) + \h}, {\b*sinh(\t) + \k});
\end{tikzpicture}
```



3.4.2 由两焦点和一点确定双曲线

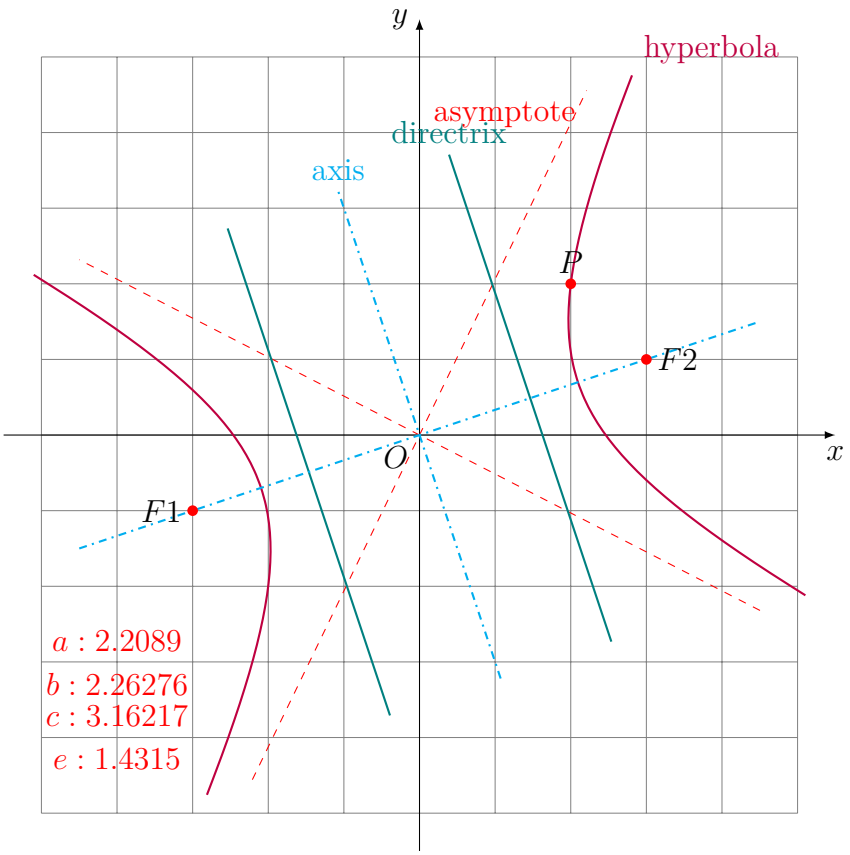
调用方式

```
`hyperbola/define = {F1,F2,P}`: define a hyperbola
↪ with two foci and a point.
`hyperbola/domain=t1:t2`: set the domain for
↪ parametric equation: x = a*cosh(t), y =
↪ b*sinh(t).
`hyperbola`: creates the hyperbola path.
`hyperbola/directrix/scale=k`: set the scale of
↪ directrices.
`hyperbola/directrix`: create the directrices.
`hyperbola/axis/scale=k`: set the scale of axes.
`hyperbola/axis`: create the axes.
`hyperbola/a`, `hyperbola/b`, `hyperbola/c`,
↪ `hyperbola/e`: hyperbola semimajor-axis,
↪ semiminor-axis, linear eccentricity,
↪ eccentricity.
```

示例

```
\begin{tikzpicture}
  \axes(-5:5, -5:5);

  \coordinate (F1) at (-3,-1);
  \coordinate (F2) at (3,1);
  \coordinate (P) at (2,2);

  \tikzset{
    hyperbola/define={F1,F2,P},
    hyperbola/domain=-1.25:1.25,
  }
```

```
    \draw [purple, thick,hyperbola] node [above right]
    ↪ {hyperbola};
    \draw [red, dashed, asymptote] node [below left]
    ↪ {asymptote};
    \draw [teal, thick,hyperbola/directrix/scale=1.
    ↪ 5,hyperbola/directrix] node [above] {directrix};
    \draw [cyan, thick, dashdotted,
    ↪ hyperbola/axis/scale=1.5,hyperbola/axis] node
    ↪ [above] {axis};

    % hyperbola parameters
    \draw[red] (-4,-3) node[above]
    ↪ {$a:{\pgfkeysvalueof{/tikz/hyperbola/a}}$};
    \draw[red] (-4,-3) node[below]
    ↪ {$b:{\pgfkeysvalueof{/tikz/hyperbola/b}}$};
    \draw[red] (-4,-4) node[above]
    ↪ {$c:{\pgfkeysvalueof{/tikz/hyperbola/c}}$};
    \draw[red] (-4,-4) node[below]
    ↪ {$e:{\pgfkeysvalueof{/tikz/hyperbola/e}}$};

    \foreach \p/\placement in {F1/left,F2/right,P/above}{
      \fill[red] (\p) circle (2pt);
      \draw (\p) node[\placement] {$\p$};
    }

  \end{tikzpicture}
```



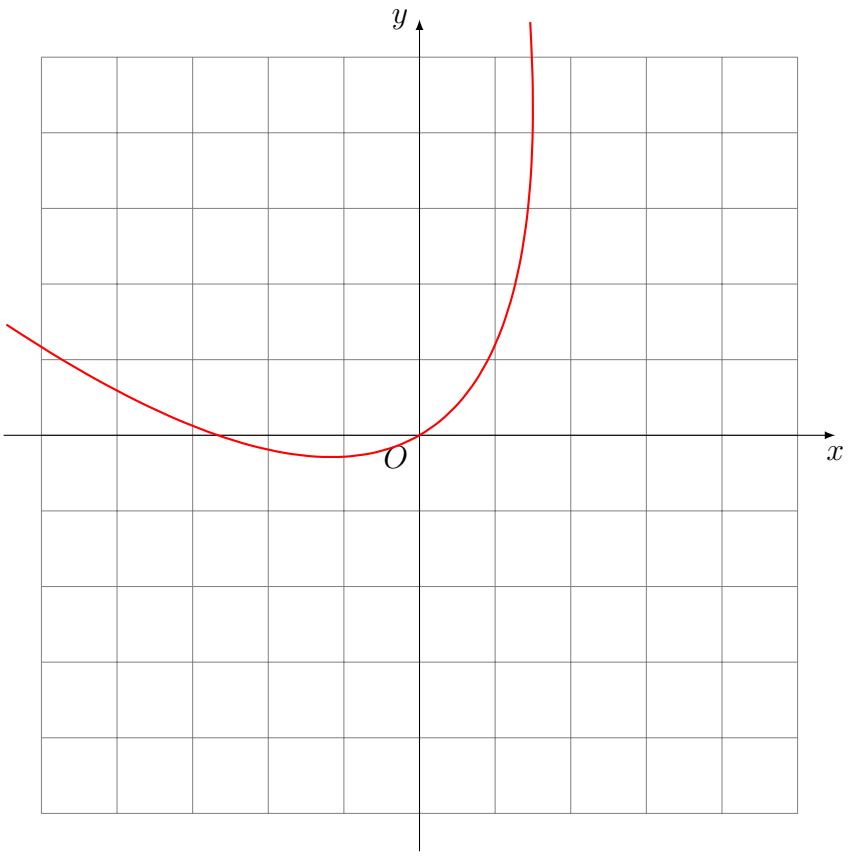
3.5 根据圆锥曲线方程绘制图形

调用方式

```
`conic/define = {a,b,c,d,e,f}`: defines a conic with
↪  equation coefficients, i.e. ax^2 + bxy + cy^2 + dx
↪  + ey + f = 0.
`conic`: creates the conic path.
```

示例

```
\begin{tikzpicture}
  \tikzset {
    conic/define={3,2*sqrt(3),1,8,-8*sqrt(3),0},
  }
  \axes (-5:5,-5:5);
  \draw[red,thick,conic];
\end{tikzpicture}
```



3.6 过椭圆外一点作椭圆的切线 Tangents to an Ellipse

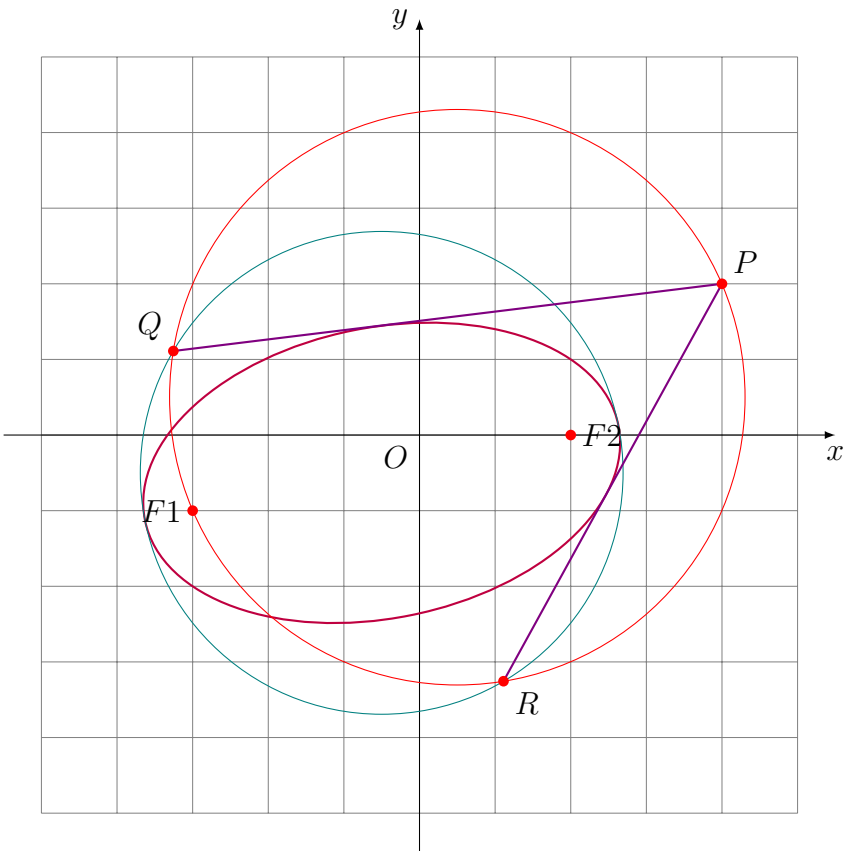
过椭圆外一点作椭圆的切线的方法有: 利用椭圆的光学性质作图; 或者根据射影几何理论, 采用极点极线法. 此处采用前者.

方法 1:

1. 作出椭圆的辅助圆 (以椭圆中心为圆心, 以长半轴长 a 为半径的圆);
2. 作出一个以 PF_1 或 PF_2 为直径的圆;
3. 两圆交于 Q, R, 则直线 PQ, PR 即为所求.

```
\begin{tikzpicture}
  \axes(-5:5, -5:5);

  \coordinate (F1) at (-3,-1);
  \coordinate (F2) at (2,0);
  \coordinate (P1) at (2,1);
  \coordinate (P) at (4,2);

  % 绘制椭圆
  \tikzset{ellipse/define ={F1,F2,P1}, }
  \draw [purple, thick, ellipse];

  % 获取存储的坐标（椭圆中心）
  \coordinate (C) at
  ↪  (\pgfkeysvalueof{/tikz/ellipse/xcenter} cm,
    \pgfkeysvalueof{/tikz/ellipse/ycenter} cm);

  % 椭圆长轴的顶点
  \coordinate (A) at
    ($(C)+(\pgfkeysvalueof{/tikz/ellipse/angle}:\
    ↪  pgfkeysvalueof{/tikz/ellipse/a} cm)$);

  % 作椭圆的辅助圆
  \draw[teal,circle={C,A}];

  % 作以 PF1 为直径的圆
  \coordinate (M) at ($(P)!.5!(F1)$);
```

```
    \draw[red,circle={M,P}];

    % 求两圆的交点（存储在 cc1 和 cc2 中）
    \tikzset{circle-circle={C,A,M,P},}
    \coordinate (Q) at (cc1);
    \coordinate (R) at (cc2);

    % 作出切线
    \draw[violet,thick] (P) -- (Q) (P) -- (R);

    \foreach \p/\placement in {F1/left,F2/right,
    P/above right,Q/above left,R/below right}{
      \fill[red] (\p) circle (2pt);
      \draw (\p) node[\placement] {$\p$};
    }

  \end{tikzpicture}
```



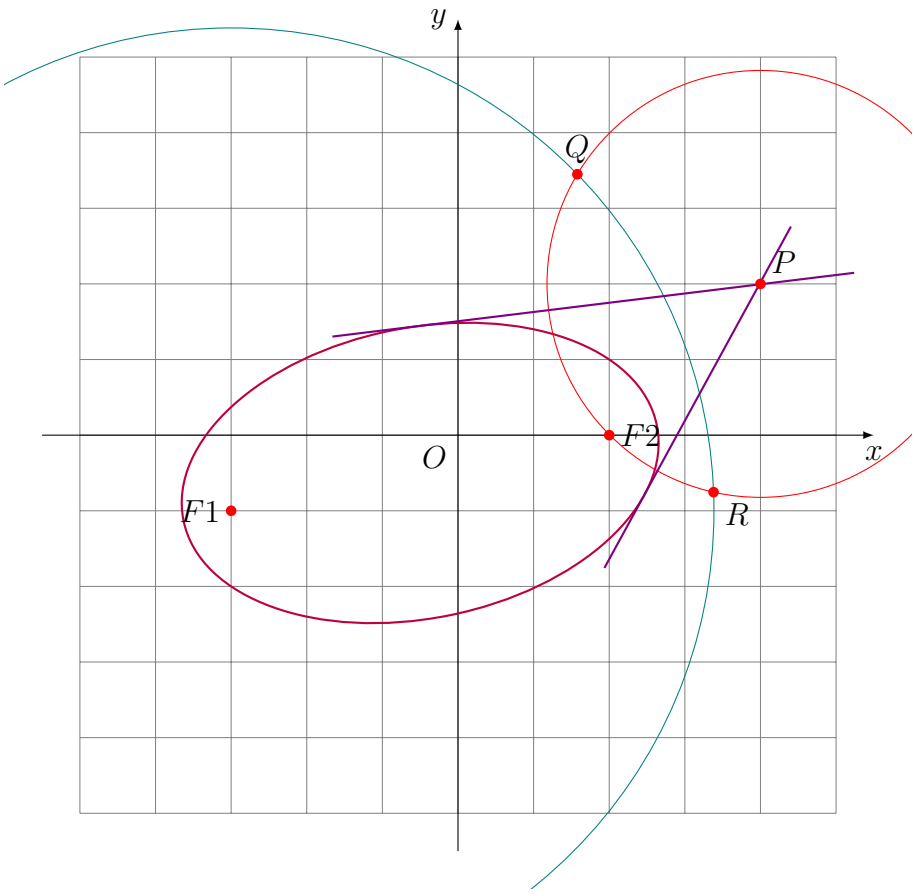
方法 2:

1. 作出一个以 F_1 为圆心, $2a$ 为半径的圆;
2. 作出一个以 F_2 为圆心, PF_2 为半径的圆;
3. 两圆交于 Q, R, 则 F_2Q, F_2R 的中垂线即为所求.

```
\begin{tikzpicture}
  \clip (-6,-6) rectangle (6,6);
  \axes(-5:5, -5:5);

  \coordinate (F1) at (-3,-1);
  \coordinate (F2) at (2,0);
  \coordinate (P1) at (2,1);
  \coordinate (P) at (4,2);
```

```
% 绘制椭圆
\tikzset{ellipse/define ={F1,F2,P1}, }
\draw [purple, thick, ellipse];

% 作以 F1 为圆心, 2a 为半径的圆
\coordinate (A) at
  ($(F1)+(\pgfkeysvalueof{/tikz/ellipse/angle}:2*\
  ↪  pgfkeysvalueof{/tikz/ellipse/a} cm)$);
\draw[teal,circle={F1,A}];

% 作以 P 为圆心, PF2 为半径的圆
\draw[red,circle={P,F2}];

% 求两圆的交点 (存储在 cc1 和 cc2 中)
\tikzset{circle-circle={F1,A,P,F2},}
\coordinate (Q) at (cc1);
\coordinate (R) at (cc2);

% 作出 F2Q, F2R 的中垂线即为椭圆的切线
\draw[violet,thick,perpendicular bisector={F2, Q}];
\draw[violet,thick,start modifier=-2cm,perpendicular
↪  bisector={F2, R}];

\foreach \p/\placement in {F1/left,F2/right,
P/above right,Q/above,R/below right}{
  \fill[red] (\p) circle (2pt);
  \draw (\p) node[\placement] {$\p$};
}
```

```
\end{tikzpicture}
```



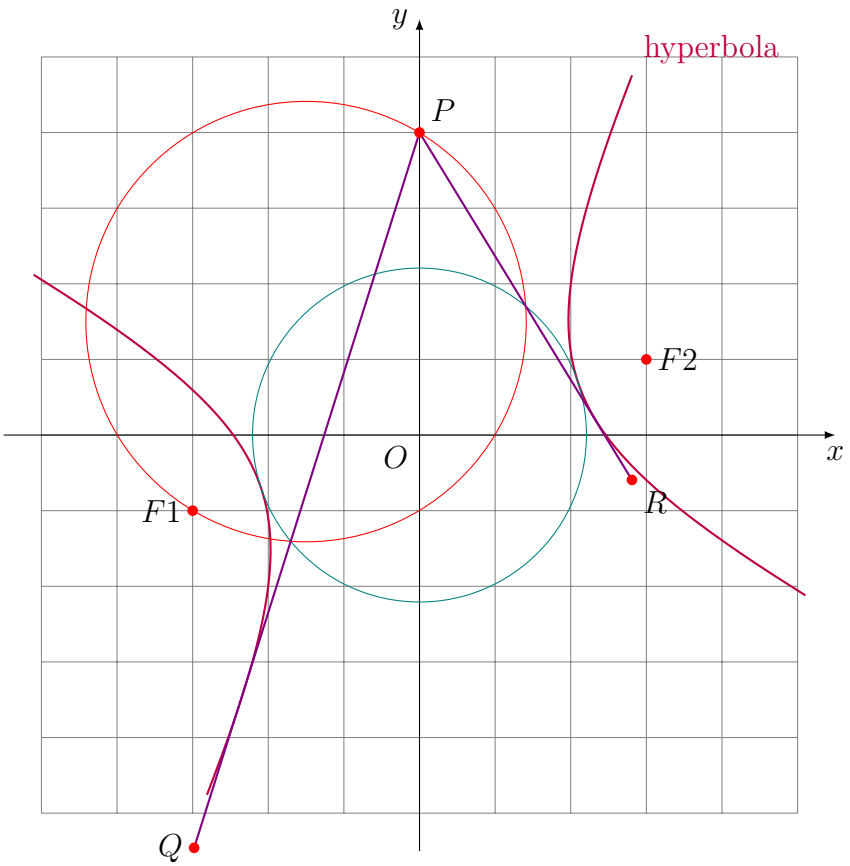
3.7 过双曲线外一点作双曲线的切线 Tangents to a Hyperbola

类似椭圆的切线作法, 对于双曲线同样有下面的方法:

```
\begin{tikzpicture}
  \axes(-5:5, -5:5);

  \coordinate (F1) at (-3,-1);
  \coordinate (F2) at (3,1);
```

```
\coordinate (P1) at (2,2);
\coordinate (P) at (0,4);

\tikzset{
  hyperbola/define={F1,F2,P1},
  hyperbola/domain=-1.25:1.25,
}

% 绘制双曲线
\draw [purple, thick,hyperbola] node [above right]
↪  {hyperbola};

% 获取存储的坐标（中心）
\coordinate (C) at
↪  (\pgfkeysvalueof{/tikz/hyperbola/xcenter} cm,
  \pgfkeysvalueof{/tikz/hyperbola/ycenter} cm);

% 双曲线的顶点
\coordinate (A) at
  ($(C)+(\pgfkeysvalueof{/tikz/hyperbola/angle}:\
  ↪  pgfkeysvalueof{/tikz/hyperbola/a} cm)$);

% 作辅助圆
\draw[teal,circle={C,A}];

% 作以 PF1 为直径的圆
\coordinate (M) at ($(P)!.5!(F1)$);
\draw[red,circle={M,P}];

% 求两圆的交点（存储在 cc1 和 cc2 中）
```

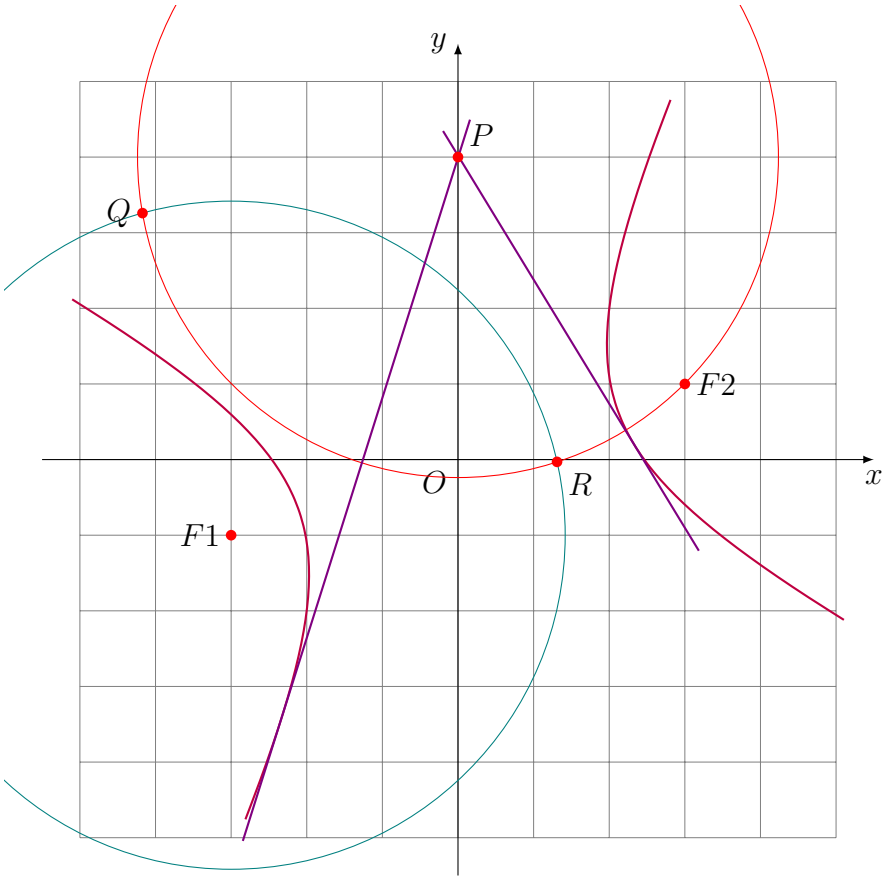
```
    \tikzset{circle-circle={C,A,M,P},}
    \coordinate[affine={P,cc1,1.75}] (Q);
    \coordinate[affine={P,cc2,2}] (R);

    % 作出切线
    \draw[violet,thick] (P) -- (Q) (P) -- (R);

    \foreach \p/\placement in {F1/left,F2/right,
    P/above right,Q/left,R/below right}{
      \fill[red] (\p) circle (2pt);
      \draw (\p) node[\placement] {$\p$};
    }

  \end{tikzpicture}
```




```
\begin{tikzpicture}
  \clip (-6,-6) rectangle (6,6);
  \axes(-5:5, -5:5);

  \coordinate (F1) at (-3,-1);
  \coordinate (F2) at (3,1);
  \coordinate (P1) at (2,2);
  \coordinate (P) at (0,4);

  \tikzset{
    hyperbola/define={F1,F2,P1},
    hyperbola/domain=-1.25:1.25,
```

```
}

% 绘制双曲线
\draw [purple, thick,hyperbola];

% 作以 F1 为圆心, 2a 为半径的圆
\coordinate (A) at
  ($(F1)+(\pgfkeysvalueof{/tikz/hyperbola/angle}:2*\
  ↪ pgfkeysvalueof{/tikz/hyperbola/a} cm)$);
\draw[teal,circle={F1,A}];

% 作以 P 为圆心, PF2 为半径的圆
\draw[red,circle={P,F2}];

% 求两圆的交点 (存储在 cc1 和 cc2 中)
\tikzset{circle-circle={F1,A,P,F2},}
\coordinate (Q) at (cc1);
\coordinate (R) at (cc2);

% 作出 F2Q, F2R 的中垂线即为切线
\draw[violet,thick,end modifier=10cm,perpendicular
↪ bisector={F2, Q}];
\draw[violet,thick,end modifier=6.5cm,perpendicular
↪ bisector={F2, R}];

\foreach \p/\placement in {F1/left,F2/right,
P/above right,Q/left,R/below right}{
  \fill[red] (\p) circle (2pt);
  \draw (\p) node[\placement] {$\p$};
}
```

```
\end{tikzpicture}
```



3.8 过抛物线外一点作抛物线的切线 Tangents to a Parabola

抛物线的切线作法如下:

1. 以点 P 为圆心, PF 为半径作圆, 交准线将于 Q, R;
2. FQ, FR 的中垂线即为所求.

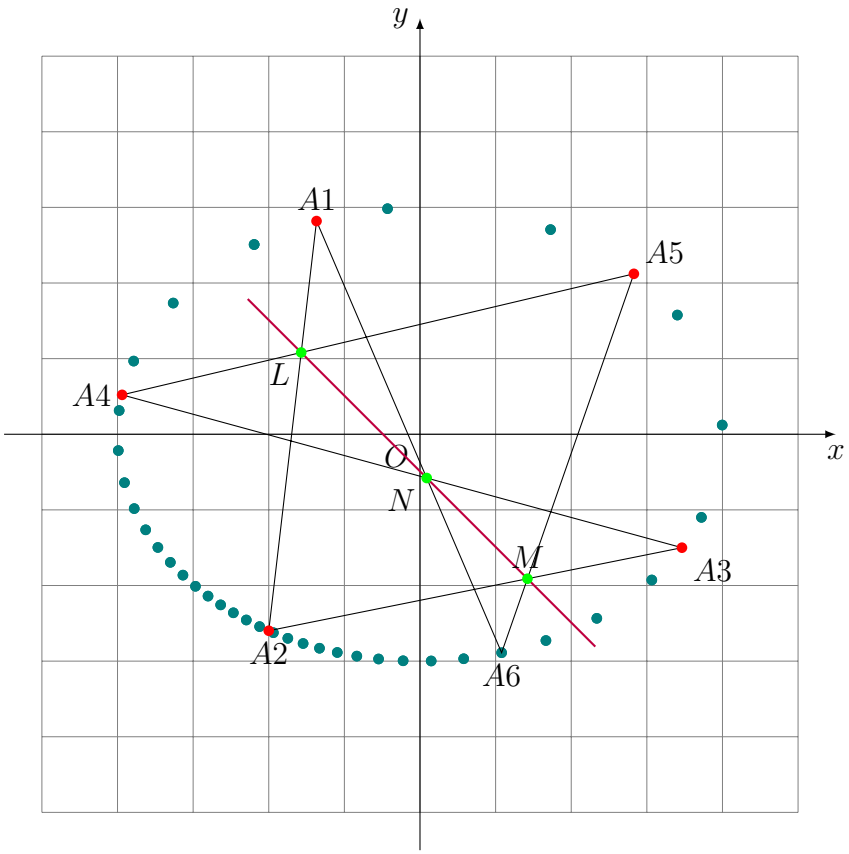
3.9 过五点作圆锥曲线 Five Points Determine a Conic

Braikenridge-Maclaurin 定理指出，如果通过六边形的相对侧的三对线的三个交点位于一条直线上，则六边形的六个顶点位于圆锥上. 这实际上是 Pascal 定理的逆定理.

示例

```
% Braikenridge-Maclaurin Construction
% the converse to Pascal's theorem
\begin{tikzpicture}
  \clip (-6,-6) rectangle (6,6);
  \axes(-5:5,-5:5);

  % 为方便起见，使用了椭圆参数方程来生成 5 个点
  \tikzmath{
    \a = 4;
    \b = 3;
  }
  \foreach \i/\angle in {1/110,2/240,3/330,4/170,5/45}{
    \coordinate (A\i) at
    ↪  ({\a*cos(\angle)},{\b*sin(\angle)});
  }

  % 根据 Pascal 定理求得一对对边的交点 L
  \coordinate[intersect={A1,A2,A4,A5}] (L);

  % 过 L 作任意直线 a，这里使用了单位向量来构造任意直线
  % 求直线 a 与 A2A3 的交点 M
  % 求直线 a 与 A3A4 的交点 N
  % 第 6 点 A6 是 A5M 与 A1N 的交点
  \foreach \angle in {0,5,...,360}{
```

```
    \coordinate (U) at ($(L)+(\angle:1)$);
    \coordinate[intersect={A2,A3,L,U}] (M);
    \coordinate[intersect={A3,A4,L,U}] (N);
    \coordinate[intersect={A5,M,A1,N}] (A6);
    \fill[teal] (A6) circle (2pt);

    % 绘制中间过程
    \ifthenelse{\angle=315}{%if-part
      \draw[thick,purple,start modifier=-1cm,end
      ↪ modifier=5.5cm,extend={L,M}];
      \draw (A1) -- (A2) -- (A3) -- (A4) -- (A5) --(A6)
      ↪ -- cycle;
      \draw (A6) node[below] {$A6$};
      \foreach \p/\placement in {L/below
      ↪ left,M/above,N/below left}{
        \fill[green] (\p) circle (2pt);
        \draw (\p) node[\placement] {$\p$};
      }
    }{%else-part
    }
  }

  \foreach \i/\placement in {1/above,2/below,3/below
  ↪ right,4/left,5/above right}{
    \fill[red] (A\i) circle (2pt);
    \draw (A\i) node[\placement] {$A\i$};
  }
\end{tikzpicture}
```



3.10 与五条直线相切的圆锥曲线 Five Tangents Determine a Conic

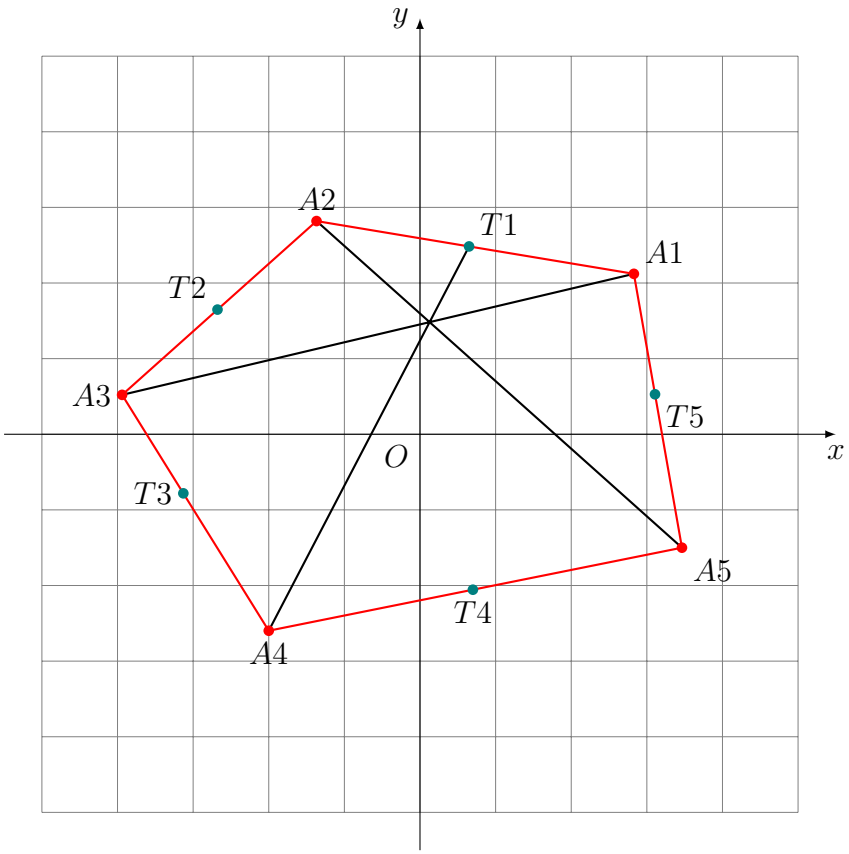
下面的作图采用 Brianchon 定理中退化的外切六边形, 求出各边的切点, 然后根据 5 点确定一个圆锥曲线 (略).

示例

```
% the converse to Brianchon's theorem
\begin{tikzpicture}
  \clip (-6,-6) rectangle (6,6);
  \axes(-5:5,-5:5);
```

```
% 为方便起见，使用了椭圆参数方程来生成 5 个点
% 用这 5 个点产生 5 条切线
\tikzmath{
  \a = 4;
  \b = 3;
}
\foreach \i/\angle in {1/45,2/110,3/170,4/240,5/330}{
  \coordinate (A\i) at
  ↪  ({\a*cos(\angle)},{\b*sin(\angle)});
}

% 利用 Brianchon 定理中退化的外接六边形（只有 5 边，其
↪  中一边为切线），
% 求各直线上的切点
\coordinate[intersect={A1,A3,A2,A5}] (B1);
\coordinate[intersect={A1,A2,A4,B1}] (T1);

\coordinate[intersect={A1,A3,A2,A4}] (B2);
\coordinate[intersect={A2,A3,A5,B2}] (T2);

\coordinate[intersect={A2,A4,A3,A5}] (B3);
\coordinate[intersect={A3,A4,A1,B3}] (T3);

\coordinate[intersect={A3,A5,A1,A4}] (B4);
\coordinate[intersect={A4,A5,A2,B4}] (T4);

\coordinate[intersect={A1,A4,A2,A5}] (B5);
\coordinate[intersect={A5,A1,A3,B5}] (T5);
```

```
    \draw[red,thick] (A1) -- (A2) -- (A3) -- (A4) -- (A5)
    ↪  -- cycle;
    \draw[thick] (A1) -- (A3)  (A5) -- (A2) (A4) -- (T1);

    \foreach \i/\placement in {1/above
    ↪  right,2/above,3/left,4/below,5/below right}{
      \fill[red] (A\i) circle (2pt);
      \draw (A\i) node[\placement] {$A\i$};
    }

    \foreach \i/\placement in {1/above right,2/above
    ↪  left,3/left,4/below,5/below right}{
      \fill[teal] (T\i) circle (2pt);
      \draw (T\i) node[\placement] {$T\i$};
    }

  \end{tikzpicture}
```


y
x
O
A2
T1
A1
T2
A3
T5
T3
A5
T4
A4

附录 A　两直线的交点

求解两直线交点的方程 [4]:

$$\begin{vmatrix} x & y & 1 \\ x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{vmatrix} = 0$$
$$\begin{vmatrix} x & y & 1 \\ x_3 & y_3 & 1 \\ x_4 & y_4 & 1 \end{vmatrix} = 0$$

注意, 两个方程的系数都是行列式, 解得:

$$x = \frac{\begin{vmatrix} \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} & \begin{vmatrix} x_1 & 1 \\ x_2 & 1 \end{vmatrix} \\ \begin{vmatrix} x_3 & y_3 \\ x_4 & y_4 \end{vmatrix} & \begin{vmatrix} x_3 & 1 \\ x_4 & 1 \end{vmatrix} \end{vmatrix}}{\begin{vmatrix} \begin{vmatrix} x_1 & 1 \\ x_2 & 1 \end{vmatrix} & \begin{vmatrix} y_1 & 1 \\ y_2 & 1 \end{vmatrix} \\ \begin{vmatrix} x_3 & 1 \\ x_4 & 1 \end{vmatrix} & \begin{vmatrix} y_3 & 1 \\ y_4 & 1 \end{vmatrix} \end{vmatrix}} = \frac{\begin{vmatrix} \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} & x_1 - x_2 \\ \begin{vmatrix} x_3 & y_3 \\ x_4 & y_4 \end{vmatrix} & x_3 - x_4 \end{vmatrix}}{\begin{vmatrix} x_1 - x_2 & y_1 - y_2 \\ x_3 - x_4 & y_3 - y_4 \end{vmatrix}}$$

$$y = \frac{\begin{vmatrix} \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} & \begin{vmatrix} y_1 & 1 \\ y_2 & 1 \end{vmatrix} \\ \begin{vmatrix} x_3 & y_3 \\ x_4 & y_4 \end{vmatrix} & \begin{vmatrix} y_3 & 1 \\ y_4 & 1 \end{vmatrix} \end{vmatrix}}{\begin{vmatrix} \begin{vmatrix} x_1 & 1 \\ x_2 & 1 \end{vmatrix} & \begin{vmatrix} y_1 & 1 \\ y_2 & 1 \end{vmatrix} \\ \begin{vmatrix} x_3 & 1 \\ x_4 & 1 \end{vmatrix} & \begin{vmatrix} y_3 & 1 \\ y_4 & 1 \end{vmatrix} \end{vmatrix}} = \frac{\begin{vmatrix} \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} & y_1 - y_2 \\ \begin{vmatrix} x_3 & y_3 \\ x_4 & y_4 \end{vmatrix} & y_3 - y_4 \end{vmatrix}}{\begin{vmatrix} x_1 - x_2 & y_1 - y_2 \\ x_3 - x_4 & y_3 - y_4 \end{vmatrix}}$$

进一步化简得到[1]:

$$x = \frac{(x_1y_2 - y_1x_2)(x_3 - x_4) - (x_1 - x_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}$$
$$y = \frac{(x_1y_2 - y_1x_2)(y_3 - y_4) - (y_1 - y_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}$$

上述方法给出的交点坐标公式在 TikZ 环境中的计算稳定性不够好, 经常出现 `Dimension too large` 错误, 究其原因是分母可能有时会比较小. 下面给出一个计算更稳定的公式.

我们可以给出两条直线的参数方程:

直线 L_1 的方程:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + s \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \end{bmatrix}$$

直线 L_2 的方程:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_3 \\ y_3 \end{bmatrix} + t \begin{bmatrix} x_4 - x_3 \\ y_4 - y_3 \end{bmatrix}$$

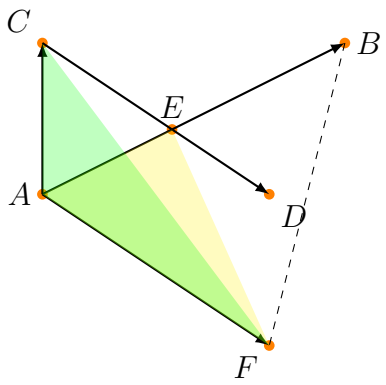
[1] https://en.wikipedia.org/wiki/Line-line_intersection

可以解出 s, t:

$$s = \frac{\begin{vmatrix} x_1 - x_3 & x_3 - x_4 \\ y_1 - y_3 & y_3 - y_4 \end{vmatrix}}{\begin{vmatrix} x_1 - x_2 & x_3 - x_4 \\ y_1 - y_2 & y_3 - y_4 \end{vmatrix}}$$

$$t = \frac{\begin{vmatrix} x_1 - x_3 & x_1 - x_2 \\ y_1 - y_3 & y_1 - y_2 \end{vmatrix}}{\begin{vmatrix} x_1 - x_2 & x_3 - x_4 \\ y_1 - y_2 & y_3 - y_4 \end{vmatrix}}$$

我们也可从几何的角度来分析:



$$\overrightarrow{AE} = s\overrightarrow{AB}$$

$$\begin{aligned} s &= \frac{S_{\triangle AEF}}{S_{\triangle ABF}} \\ &= \frac{S_{\triangle ACF}}{S_{\triangle ABF}} \\ &= \frac{\overrightarrow{AF} \times \overrightarrow{AC}}{\overrightarrow{AF} \times \overrightarrow{AB}} \\ &= \frac{\overrightarrow{CD} \times \overrightarrow{AC}}{\overrightarrow{CD} \times \overrightarrow{AB}} \end{aligned}$$

为了保证数值计算的稳定性, 可以对下面的方程进行列主元消元法求解:

$$x_1 + s(x_2 - x_1) = x3 + t(x_4 - x3)$$
$$y_1 + s(y_2 - y_1) = y3 + t(y_4 - y3)$$

附录 B　通径 Latus Rectum

通径 (latus rectum) 亦称” 正通径”,” 首通径”,” 直焦弦”,” 主焦弦”,” 正焦弦”. 过圆锥曲线的焦点且与过焦点的轴垂直的弦称为通径, 清代明安图《割环密率捷法》中, 称圆的直径为通径.

The distance p from the focus to the conic section directrix of a conic section. The following table gives the focal parameter for the different types of conics, where a is the semimajor axis, b is the semiminor axis, c is the distances from the origin to the focus, and e is the eccentricity.

表 B.1: Focal Parameters of Conics

Conic	e	p
Ellipse	$0 < e < 1$	$p = \dfrac{b^2}{c}$
Parabola	$e = 1$	$p = 2a$
Hyperbola	$e > 1$	$p = \dfrac{b^2}{c}$

B.1　椭圆的通径与焦准距

连接椭圆上任意两点的线段叫作这个**椭圆的弦**, 通过焦点的弦叫作这个椭圆的**焦点弦** (所以椭圆的长轴也是焦点弦), 和长轴垂直的焦点弦叫作这个椭圆的**通径 (正焦弦)**. 连接椭圆上任意一点与一个焦点的线段 (或这线段的长) 叫作椭圆在这点的**焦半径**, 椭圆上任意一点有两条焦半径.

设椭圆的方程为

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1, (a > b > 0)$$

其通径的长为 $\frac{2b^2}{a}$, 或 $2ep$, 其中: a 为**半长轴长**, b 为**半短轴长**, e 为椭圆的**离心率**, p 为椭圆的**焦准距**.

焦点与相应准线的距离称为椭圆的**焦准距**, 也叫**焦参数**.

$$p = \frac{a^2}{c} - c = \frac{a^2 - c^2}{c} = \frac{b^2}{c}$$

B.2 抛物线的通径与焦准距

经过抛物线的焦点, 作一条垂直于它的对称轴的直线, 这直线与抛物线有两个交点, 这两个交点之间的线段叫做**抛物线的通径**.

抛物线的方程:

$$y^2 = 2px$$

抛物线的通径长为 $2p$, 其中: 抛物线的**焦准距**长为 p.

B.3 双曲线的通径与焦准距

过双曲线的焦点与双曲线的实轴垂直的直线被双曲线截得的线段的长, 称为**双曲线的通径**.

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1, (a > 0, b > 0)$$

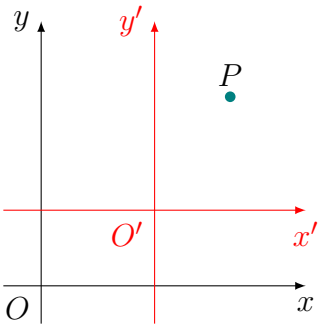
其通径长 $\frac{2b^2}{a}$, 或 $2ep$, 其中: a 为**半实轴长**, b 为**半虚轴长**, e 为双曲线的**离心率**, p 为双曲线的**焦准距**.

附录 C　圆锥曲线的变换 Conic Transformations

C.1　坐标变换 Coordinate Transformations

下面讨论坐标轴平移和旋转后坐标的变化.

C.1.1　Translation

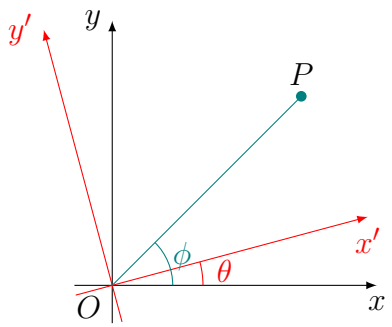


$$\begin{cases} x' = x - dx \\ y' = y - dy \end{cases}$$

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & -d_x \\ 0 & 1 & -d_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}$$

C.1.2 Rotation



$$\begin{cases} x = r\cos\phi \\ y = r\sin\phi \end{cases}$$

$$\begin{cases} x' = r\cos(\phi-\theta) \\ y' = r\sin(\phi-\theta) \end{cases}$$

$$\begin{cases} x' = x\cos\theta + y\sin\theta \\ y' = y\cos\theta - x\sin\theta \end{cases}$$

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}$$

C.2 圆锥曲线的变换

C.2.1 圆锥曲线方程的一般形式

设二次曲线的方程为:

$$ax^2 + bxy + cy^2 + dx + ey + f = 0$$

记

$$Q = \begin{pmatrix} a & b/2 & d/2 \\ b/2 & c & e/2 \\ d/2 & e/2 & f \end{pmatrix}$$

$$\mathbf{x} = (x, y, 1)^T$$

则:

$$\mathbf{x}^T Q \mathbf{x} = 0$$

C.2.2 圆锥曲线的切线方程

设 $\mathbf{x}_0$ 是圆锥曲线上的一点, 则过该点的切线方程是:

$$\mathbf{x}_0^T Q \mathbf{x} = 0$$

C.2.3 圆锥曲线的中心

设 Q_{31}, Q_{32}, Q_{33} 是矩阵 Q 的代数余子式, 如果 $Q_{33} = 0$, 则曲线是抛物线, 其中心在无穷远处; 否则曲线的中心是 $\left(\frac{Q_{31}}{Q_{33}}, \frac{Q_{32}}{Q_{33}}\right)$.

C.2.4 将圆锥曲线方程从一般形式化为标准形式

因为抛物线是没有中心的, 对于任意圆锥曲线, 可以先进行旋转, 消除交叉项 xy, 然后再处理.

旋转坐标系

首先是旋转, 消除交叉项 xy, 旋转角度为 $\theta(-\pi/4 \le \theta \le \pi/4)$, 且:

$$\cot 2\theta = \frac{1 - \tan^2 \theta}{2 \tan \theta} = \frac{a - c}{b}$$

令 $\tau = \cot 2\theta$, 则:

$$\tan^2\theta + 2\tau\tan\theta - 1 = 0$$

取绝对值最小的根, 这样保证 $-\pi/4 \le \theta \le \pi/4$, 即:

$$\tan\theta = \begin{cases} -\tau + \sqrt{\tau^2+1}, & \text{if } \tau \ge 0; \\ -\tau - \sqrt{\tau^2+1}, & \text{otherwise.} \end{cases}$$

$$t = \tan\theta$$
$$c = \cos\theta = \frac{1}{\sqrt{t^2+1}}$$
$$s = \sin\theta = \tan\theta\cos\theta = tc$$

这样无需进行三角函数运算.

构造旋转矩阵:

$$R = \begin{pmatrix} c & -s & 0 \\ s & c & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

计算 $\tilde{Q} = R^T Q R$ 就得到新的系数矩阵.

平移坐标系

然后根据系数矩阵 $\tilde{Q}$ 判断是否为抛物线, 即 $\tilde{Q}_{11} = 0$ 或 $\tilde{Q}_{22} = 0$. 如果是抛物线, 则将坐标系平移至顶点; 否则将坐标系平移至中心.

若 $\tilde{Q}_{11} = 0$ 且 $\tilde{Q}_{22} \ne 0$, 则抛物线简化为:

$$cy^2 + dx + ey + f = 0$$

其中:

$$c = \tilde{Q}_{22}$$
$$d = 2\tilde{Q}_{13}$$

$$e = 2\tilde{Q}_{23}$$
$$f = \tilde{Q}_{33}$$

则抛物线的顶点为: $\left(-\frac{e}{2c}, \frac{e^2-4cf}{4cd}\right)$.

C.2.5 五点确定圆锥曲线

设二次曲线的方程为:

$$ax^2 + bxy + cy^2 + dx + ey + f = 0$$

将 $(x_i, y_i)(i = 1, 2, \cdots, 5)$ 代入上式, 并组成线性方程组. 由于 $(a, b, c, d, e, f) \neq \mathbf{0}$, 该方程组的系数矩阵的行列式为 0, 即:

$$\det \begin{pmatrix} x^2 & xy & y^2 & x & y & 1 \\ x_1^2 & x_1y_1 & y_1^2 & x_1 & y_1 & 1 \\ x_2^2 & x_2y_2 & y_2^2 & x_2 & y_2 & 1 \\ x_3^2 & x_3y_3 & y_3^2 & x_3 & y_3 & 1 \\ x_4^2 & x_4y_4 & y_4^2 & x_4 & y_4 & 1 \\ x_5^2 & x_5y_5 & y_5^2 & x_5 & y_5 & 1 \end{pmatrix} = 0$$

化简该行列式即得到二次曲线的方程.

C.2.6 与五条直线相切的圆锥曲线

根据对偶原理, 二次曲线可以用两种方式定义, 点形式和直线形式:
点形式定义: 由点集满足的二次方程定义, 例如:

$$ax^2 + bxy + cy^2 + dxz + eyz + fz^2 = 0$$

用矩阵形式表示为:

$$(x, y, z) \begin{pmatrix} a & b/2 & d/2 \\ b/2 & c & e/2 \\ d/2 & e/2 & f \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = 0$$

称该矩阵为 Q.

设 (x_0, y_0, z_0) 是曲线上的点, 则过该点的切线方程为:

$$(x_0, y_0, z_0) \begin{pmatrix} a & b/2 & d/2 \\ b/2 & c & e/2 \\ d/2 & e/2 & f \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = 0$$

直线形式定义: 由所有与原曲线相切的直线的包络定义，用直线坐标 (l, m, n) 的二次方程表示:

$$Al^2 + Blm + Cm^2 + Dlm + Emn + Fn^2 = 0$$

同样用矩阵形式:

$$(l, m, n) \begin{pmatrix} A & B/2 & D/2 \\ B/2 & C & E/2 \\ D/2 & E/2 & F \end{pmatrix} \begin{pmatrix} l \\ m \\ n \end{pmatrix} = 0$$

称该矩阵为 Q^*. Q^* 是 Q 的伴随矩阵:

$$Q^{-1} = \frac{Q^*}{\det(Q)}.$$

我们可以根据 5 条直线的方程求出 Q^*, 再求其逆矩阵, 从而得到曲线的方程.

Find dual conic[1]:

Taking the quadratic form $p^T Ap = 0$ and remembering A is symmetric, we can also transform it:

$$p^T(AA^{-1})Ap = 0$$
$$p^T A^T A^{-1} Ap = 0$$
$$(Ap)^T A^{-1}(Ap) = 0$$

This modified form can be interpreted as the dual of the primal conic described by the quadratic form $l^{-1}l = 0$, where every $l = Ap$ that satisfies this equation is tangent to the primal conic.

Additionally, the adjugate of A can mostly be substituted for the inverse because $A^{-1} = adj(A)/det(A)$.

[1] https://math.stackexchange.com/questions/460404/finding-dual-conic

附录 D　源代码

D.1　绘图包

```
\ProvidesFile{tikzlibraryeuclidea.code.tex}[2025/03/19
↪  v1.6.0 A tikz library for Euclidean geometry
↪  constructions]

% \RequirePackage{xfp}

\usetikzlibrary{math,calc,quotes,mc}

% pgfmath-solution to the error:
% Package PGF Math: Could not parse input 'A' as a
↪  floating
% point number, sorry. The unreadable part was near
↪  'A'..
% https://tex.stackexchange.com/questions/455991/
↪  pgfmath-function-for-strings-and-numbers
\pgfkeys{
  /pgf/fpu/handlers/invalid number/.code = {%
    \pgfmathfloatparsenumber{3Y0.0e0]}%
  }
}
```

```
\makeatletter

% ================================================
% Caveats
% ================================================

% function 'intersectll' conflicts with siunitx package

% 在 TikZ 中，函数本身并没有直接的“返回多个值”的机制，因为 TikZ 是一种基于 TeX 的绘图语言，主要用于描述图形，
% 而不是像编程语言那样处理多返回值逻辑。不过，通过一些技巧，你可以在 TikZ 中模拟类似“返回多个值”的效果。
% https://tex.stackexchange.com/questions/510418/tikz-declare-function-with-multiple-outputs

% tikzset 中不要有空行

% 注意：计算过程是保留坐标单位（pt）的，所以存在乘除法单位的问题，首先数值始终携带单位，
% 在 calc 运算时有的需要转换为标量；将坐标转换为 pt 值，数值可能超出限值，出现
% Dimension too large 错误，在计算长度时及时进行缩小
% https://tex.stackexchange.com/questions/475556/tikz-why-is-dimension-too-large
% 具体方法是修改默认的 1cm，如：[scale=1.0,x=0.5cm,y=0.5cm]
% 注意此处的变量不要和 tikzpicture 环境重名，否则被替换掉
% triangle centers:
% https://mathworld.wolfram.com/BarycentricCoordinates.html
```

```
\tikzset{
  % specifying start and end with modifiers(see tikz
  ↪ manual 13.5)
  % commands supporting partway modifiers:
  % radical axis, perpendicular bisector, perpendicular,
  ↪ parallel
  start modifier/.initial = 0,
  % start modifier/.default = 0,
  end modifier/.initial = 1,
  % end modifier/.default = 1,
  % ========== Coordinates Transformations ==========
  % affine={A,B,k}: returns affine combination of two
  ↪ points
  % with affine ratio, i.e. A + k * ( B - A )
  affine/.style args = {#1,#2,#3}{
    insert path = {
      ($(#1)!{#3}!(#2)$)
    }
  },
  % midpoint={A,B}: returns midpoint of AB.
  midpoint/.style args = {#1,#2}{
    insert path = {
      ($(#1)!.5!(#2)$)
    }
  },
  % translate={A,B,C}: returns translation of C by
  % the vector AB, i.e. C + ( B - A )
  translate/.style args = {#1,#2,#3}{
    insert path = {
```

```
      ($(#3)+(#2)-(#1)$)
    }
  },
  % reflect={A,B,C}: reflects point C across line AB.
  reflect/.style args = {#1,#2,#3}{
    insert path = {
      let
        \p{ft} = ($(#1)!(#3)!(#2)$),% perpendicular foot
      in ($(#3)!2!(\p{ft})$)
    }
  },
  % project={A,B,C}: projects point C onto line AB.
  project/.style args = {#1,#2,#3}{
    insert path = {
      ($(#1)!(#3)!(#2)$)
    }
  },
  % circle inverse={O,A,P}: returns inverse point P with
  ↪  respect to
  % a reference circle(O,A).
  circle inverse/.style args = {#1,#2,#3}{
    insert path = {
      let
        \p{OA} = ($(#2)-(#1)$),
        \p{OP} = ($(#3)-(#1)$),
        \n{r} = {veclen(\p{OA})},
        \n{d} = {veclen(\p{OP})},
        \n1 = {scalar((\n{r}/\n{d}))},
      in ($(#1)!\n1*\n1!(#3)$)
    }
```

```
    },
    revolve/scale/.initial = 1,% angle scale
    revolve/@angle/.initial = 90,
    revolve/@argn/.initial = 0,% arguments count
    % set revolve/@angle with certain degrees or angle of
    ↪ a vector
    revolve/@set angle 1/.code args = {#1}{
      \pgfmathanglebetweenpoints
        {\pgfpoint{0cm}{0cm}}
        {\pgfpointanchor{#1}{center}}
      \pgfkeysalso{/tikz/revolve/@angle = \pgfmathresult}
      % \typeout{=========================}
      % \typeout{/tikz/revolve/@angle:\pgfkeysvalueof{/
      ↪ tikz/revolve/@angle}}
      % \typeout{=========================}
    },
    % set revolve/@angle with angle between two position
    ↪ vectors
    revolve/@set angle 2/.code args = {#1,#2}{
      \pgfmathanglebetweenpoints
        {\pgfpointanchor{#1}{center}}
        {\pgfpointanchor{#2}{center}}
      \pgfkeysalso{/tikz/revolve/@angle = \pgfmathresult}
      % \typeout{=========================}
      % \typeout{/tikz/revolve/@angle:\pgfkeysvalueof{/
      ↪ tikz/revolve/@angle}}
      % \typeout{=========================}
    },
    % set revolve/@angle with angle {A,B,C}, angle between
    ↪ two sides
```

```
% (A is apex, B is the start point, C is the end
↪ point)
revolve/@set angle 3/.code args = {#1,#2,#3}{
  \pgfmathanglebetweenlines
    {\pgfpointanchor{#1}{center}}
    {\pgfpointanchor{#2}{center}}
    {\pgfpointanchor{#1}{center}}
    {\pgfpointanchor{#3}{center}}
  \pgfkeysalso{/tikz/revolve/@angle = \pgfmathresult}
  % \typeout{=========================}
  % \typeout{/tikz/revolve/@angle:\pgfkeysvalueof{/
  ↪ tikz/revolve/@angle}}
  % \typeout{=========================}
},
% set revolve/@angle with angle between two
↪ vectors(ccw, AB and CD)
revolve/@set angle 4/.code args = {#1,#2,#3,#4}{
  \pgfmathanglebetweenlines
    {\pgfpointanchor{#1}{center}}
    {\pgfpointanchor{#2}{center}}
    {\pgfpointanchor{#3}{center}}
    {\pgfpointanchor{#4}{center}}
  \pgfkeysalso{/tikz/revolve/@angle = \pgfmathresult}
  % \typeout{=========================}
  % \typeout{/tikz/revolve/@angle:\pgfkeysvalueof{/
  ↪ tikz/revolve/@angle}}
  % \typeout{=========================}
},
revolve/angle/.code = {%
  \pgfmathfloatparsenumber{#1}
```

```
    \pgfmathfloattomacro{\pgfmathresult}{\F}{\M}{\E}
    \ifnum \F < 3%number
      \pgfmathparse{#1}
    \else
      % Compute the Number of Arguments
      \pgfutil@for\arg:=#1\do{
        \pgfmathparse{int(add(\pgfkeysvalueof{/tikz/
        ↪ revolve/@argn},1))}
        \pgfkeysalso{/tikz/revolve/@argn =
        ↪ \pgfmathresult}
      }
      \ifnum \pgfkeysvalueof{/tikz/revolve/@argn} = 1
        \tikzset{revolve/@set angle 1 = {#1}}
      \else\ifnum \pgfkeysvalueof{/tikz/revolve/@argn} =
      ↪ 2
        \tikzset{revolve/@set angle 2 = {#1}}
      \else\ifnum \pgfkeysvalueof{/tikz/revolve/@argn} =
      ↪ 3
        \tikzset{revolve/@set angle 3 = {#1}}
      \else\ifnum \pgfkeysvalueof{/tikz/revolve/@argn} =
      ↪ 4
        \tikzset{revolve/@set angle 4 = {#1}}
      \else
        \pgferror{"Incorrect number of arguments!"}
      \fi\fi\fi
      \fi
    \fi
    \pgfkeysalso{/tikz/revolve/@angle = \pgfmathresult}
  },
```

```
% revolve={A,B}: rotates point B by the angle around
↪ point A.
revolve/.style args = {#1,#2}{
  insert path = {
    let
      \n1 = {\pgfkeysvalueof{/tikz/revolve/@angle}},
      \n2 = {\pgfkeysvalueof{/tikz/revolve/scale}}
    in ($(#1)!1!\n1*\n2:(#2)$)
  }
},
% angle bisector={A,B,C}: alias for [revolve/angle={
↪ A,B,C},revolve/scale=.5,revolve={A,B}]
angle bisector/.style args = {#1,#2,#3}{
  revolve/angle={#1,#2,#3},revolve/scale=.5,revolve={
  ↪ #1,#2}
},
% erect={A,B}: alias for
↪ [revolve/angle=90,revolve={A,B}]
erect/.style args = {#1,#2}{
  revolve/angle=90,revolve={#1,#2}
},
% equilateral={A,B}: alias for
↪ [revolve/angle=60,revolve={A,B}]
equilateral/.style args = {#1,#2}{
  revolve/angle=60,revolve={#1,#2}
},
% cut a line segment of a certain length on a straight
↪ line
intercept/@length/.initial = 1cm,
intercept/scale/.initial = 1,% length scale
```

```
intercept/length/.code = {% set length by distance of
↪  segment
  \pgfutil@in@{,}{#1}
  \ifpgfutil@in@%compute segment length
    \euclidea@ComputeLength#1\euclidea@stop
    \pgfkeysalso{/tikz/intercept/@length =
    ↪  \pgfmathresult}
  \else
    \pgfkeysalso{/tikz/intercept/@length = #1}
  \fi
  % \typeout{========================}
  % \typeout{/tikz/intercept/@length:\pgfkeysvalueof{
  ↪  /tikz/intercept/@length}}
  % \typeout{========================}
},
% intercept={A,B}: intercepts a line segment(starting
% from point A) of a certain length on line AB.
intercept/.style args = {#1,#2}{
  insert path = {
    let
      \n1 =
      ↪  {\pgfkeysvalueof{/tikz/intercept/@length}},
      \n2 = {\pgfkeysvalueof{/tikz/intercept/scale}},
      \p{AB} = ($(#2)-(#1)$),
      \n{d} = {veclen(\p{AB})},
      \n3 = {scalar(\n1*\n2/\n{d})}
    in ($(#1)!\n3!(#2)$)
  }
},
```

```
% intersect={A,B,C,D}: returns the intersection
 ↪ coordinate
% of line AB and line CD.
% https://en.wikipedia.org/wiki/Line%E2%80%
 ↪ 93line_intersection
intersectll/.code args = {#1,#2,#3,#4}{
  \newdimen\xA
  \newdimen\yA
  \newdimen\xB
  \newdimen\yB
  \newdimen\xC
  \newdimen\yC
  \newdimen\xD
  \newdimen\yD
  \pgfextractx{\xA}{\pgfpointanchor{#1}{center}}
  \pgfextracty{\yA}{\pgfpointanchor{#1}{center}}
  \pgfextractx{\xB}{\pgfpointanchor{#2}{center}}
  \pgfextracty{\yB}{\pgfpointanchor{#2}{center}}
  \pgfextractx{\xC}{\pgfpointanchor{#3}{center}}
  \pgfextracty{\yC}{\pgfpointanchor{#3}{center}}
  \pgfextractx{\xD}{\pgfpointanchor{#4}{center}}
  \pgfextracty{\yD}{\pgfpointanchor{#4}{center}}
  \pgfmathsetmacro{\xa}{\xA / 1cm}
  \pgfmathsetmacro{\ya}{\yA / 1cm}
  \pgfmathsetmacro{\xb}{\xB / 1cm}
  \pgfmathsetmacro{\yb}{\yB / 1cm}
  \pgfmathsetmacro{\xc}{\xC / 1cm}
  \pgfmathsetmacro{\yc}{\yC / 1cm}
  \pgfmathsetmacro{\xd}{\xD / 1cm}
  \pgfmathsetmacro{\yd}{\yD / 1cm}
```

```
    \tikzmath{
      \a{1,1} = \yb - \ya; \a{1,2} = \xa - \xb;
      \a{2,1} = \yd - \yc; \a{2,2} = \xc - \xd;
      \b{1,1} = \xa * (\yb-\ya) - \ya * (\xb - \xa);
      \b{2,1} = \xc * (\yd-\yc) - \yc * (\xd - \xc);
    }
    \tikzset{solve={\a,\b,\c,2,1},}
    % \typeout{=========================}
    % \typeout{\a{1,1},\a{1,2}, \b{1,1}}
    % \typeout{\a{2,1},\a{2,2}, \b{2,1}}
    % \typeout{\c{1,1},\c{2,1}}
    % \typeout{=========================}
    % \coordinate (ll) at (\c{1,1},\c{2,1});
  },
  intersect/.style args = {#1,#2,#3,#4}{
    intersectll={#1,#2,#3,#4},%invoke code defined above
    insert path = {
      (\c{1,1},\c{2,1})
    }
  },
  % ========== Triangle Centers ==========
  % calculated from barycentric coordinates
  % incenter = {A,B,C}
  incenter/.style args = {#1,#2,#3}{
    insert path = {
      let
        \p1 = (#1), \p2 = (#2), \p3 = (#3),
        \p{AB} = ($(#2)-(#1)$),
        \p{BC} = ($(#3)-(#2)$),
        \p{CA} = ($(#1)-(#3)$),
```

```
        \n{a} = {veclen(\x{BC}, \y{BC})},
        \n{b} = {veclen(\x{CA}, \y{CA})},
        \n{c} = {veclen(\x{AB}, \y{AB})},
        \n{s} = {\n{a}+\n{b}+\n{c}},
        \n1 = {\n{a}/\n{s}},
        \n2 = {\n{b}/\n{s}},
        \n3 = {\n{c}/\n{s}},
      in ({\n1*\x1+\n2*\x2+\n3*\x3,\n1*\y1+\n2*\y2+\
       ↪  n3*\y3})
    }
  },
  % excenter = {A,B,C}, returns excenter opposite to the
   ↪  vertex A
  excenter/.style args = {#1,#2,#3}{
    insert path = {
      let
        \p1 = (#1), \p2 = (#2), \p3 = (#3),
        \p{AB} = ($(#2)-(#1)$),
        \p{BC} = ($(#3)-(#2)$),
        \p{CA} = ($(#1)-(#3)$),
        \n{a} = {veclen(\x{BC}, \y{BC})},
        \n{b} = {veclen(\x{CA}, \y{CA})},
        \n{c} = {veclen(\x{AB}, \y{AB})},
        \n{s} = {-\n{a}+\n{b}+\n{c}},
        \n1 = {\n{a}/\n{s}},
        \n2 = {\n{b}/\n{s}},
        \n3 = {\n{c}/\n{s}},
      in ({-\n1*\x1+\n2*\x2+\n3*\x3,-\n1*\y1+\n2*\y2+\
       ↪  n3*\y3})
    }
```

```
},
% circumcenter = {A,B,C}
circumcenter/.style args = {#1,#2,#3}{
  insert path = {
    let
      \p1 = (#1), \p2 = (#2), \p3 = (#3),
      \p{AB} = ($(#2)-(#1)$),
      \p{BC} = ($(#3)-(#2)$),
      \p{CA} = ($(#1)-(#3)$),
      \n{a} = {veclen(\x{BC}, \y{BC})},
      \n{b} = {veclen(\x{CA}, \y{CA})},
      \n{c} = {veclen(\x{AB}, \y{AB})},
      \n{m} = {max(max(\n{a},\n{b}),\n{c})},
      \n{a} = {\n{a}/\n{m}},
      \n{a} = {\n{a}*\n{a}},
      \n{b} = {\n{b}/\n{m}},
      \n{b} = {\n{b}*\n{b}},
      \n{c} = {\n{c}/\n{m}},
      \n{c} = {\n{c}*\n{c}},
      \n1 = {\n{a}*(\n{b}+\n{c}-\n{a})},
      \n2 = {\n{b}*(\n{c}+\n{a}-\n{b})},
      \n3 = {\n{c}*(\n{a}+\n{b}-\n{c})},
      \n{s} = {\n1+\n2+\n3},
      \n1 = {\n1/\n{s}},
      \n2 = {\n2/\n{s}},
      \n3 = {\n3/\n{s}},
    in ({\n1*\x1+\n2*\x2+\n3*\x3,\n1*\y1+\n2*\y2+\
    ↪  n3*\y3})
  }
},
```

```
% orthocenter = {A,B,C}
orthocenter/.style args = {#1,#2,#3}{
  insert path = {
    let
      \p1 = (#1), \p2 = (#2), \p3 = (#3),
      \p{AB} = ($(#2)-(#1)$),
      \p{BC} = ($(#3)-(#2)$),
      \p{CA} = ($(#1)-(#3)$),
      \n{a} = {veclen(\x{BC}, \y{BC})},
      \n{b} = {veclen(\x{CA}, \y{CA})},
      \n{c} = {veclen(\x{AB}, \y{AB})},
      \n{m} = {max(max(\n{a},\n{b}),\n{c})},
      \n{a} = {\n{a}/\n{m}},
      \n{a} = {\n{a}*\n{a}},
      \n{b} = {\n{b}/\n{m}},
      \n{b} = {\n{b}*\n{b}},
      \n{c} = {\n{c}/\n{m}},
      \n{c} = {\n{c}*\n{c}},
      \n{a2} = {\n{b}+\n{c}-\n{a}},
      \n{b2} = {\n{c}+\n{a}-\n{b}},
      \n{c2} = {\n{a}+\n{b}-\n{c}},
      \n1 = {\n{c2}*\n{b2}},
      \n2 = {\n{a2}*\n{c2}},
      \n3 = {\n{b2}*\n{a2}},
      \n{s} = {\n1+\n2+\n3},
      \n1 = {\n1/\n{s}},
      \n2 = {\n2/\n{s}},
      \n3 = {\n3/\n{s}},
    in ({\n1*\x1+\n2*\x2+\n3*\x3,\n1*\y1+\n2*\y2+\
    ↪  n3*\y3})
```

```
    }
  },
  % centroid = {A,B,C}
  centroid/.style args = {#1,#2,#3}{
    insert path = {
      let
        \p1 = (#1), \p2 = (#2), \p3 = (#3),
      in ({(\x1+\x2+\x3)/3},{(\y1+\y2+\y3)/3})
    }
  },
  % nine-pint center = {A,B,C}
  nine-point center/.style args = {#1,#2,#3}{
    insert path = {
      let
        \p1 = (#1), \p2 = (#2), \p3 = (#3),
        \p{AB} = ($(#2)-(#1)$),
        \p{BC} = ($(#3)-(#2)$),
        \p{CA} = ($(#1)-(#3)$),
        \n{a} = {veclen(\x{BC}, \y{BC})},
        \n{b} = {veclen(\x{CA}, \y{CA})},
        \n{c} = {veclen(\x{AB}, \y{AB})},
        \n{m} = {max(max(\n{a},\n{b}),\n{c})},
        \n{a} = {\n{a}/\n{m}},
        \n{a} = {\n{a}*\n{a}},
        \n{b} = {\n{b}/\n{m}},
        \n{b} = {\n{b}*\n{b}},
        \n{c} = {\n{c}/\n{m}},
        \n{c} = {\n{c}*\n{c}},
        \n1 = {\n{a}*(\n{b}+\n{c})-(\n{b}-\n{c})*(\n{b}
        ↪  -\n{c})},
```

```
        \n2 = {\n{b}*(\n{c}+\n{a})-(\n{c}-\n{a})*(\n{c}
        ↪  -\n{a})},
        \n3 = {\n{c}*(\n{a}+\n{b})-(\n{a}-\n{b})*(\n{a}
        ↪  -\n{b})},
        \n{s} = {\n1+\n2+\n3},
        \n1 = {\n1/\n{s}},
        \n2 = {\n2/\n{s}},
        \n3 = {\n3/\n{s}},
      in ({\n1*\x1+\n2*\x2+\n3*\x3,\n1*\y1+\n2*\y2+\
      ↪  n3*\y3})
    }
},
% ========== Circle Operations ==========
% circle = {O,A}, creates circle with the center (O)
↪  through A
circle/.style args = {#1,#2}{
  insert path = {
    let
      \p{OA} = ($(#2)-(#1)$),
    in (#1) circle ({veclen(\p{OA})})
  }
},
% tagent point = {O,A,P} NOTE: deprecated
% O,A: center of circle and an abitary point on the
↪  circle
% P: a point outside the circle
tangent point/.style args = {#1,#2,#3}{
  insert path = {
    let
      \p{OA} = ($(#2)-(#1)$), % 半径
```

```
        \p{OP} = ($(#3)-(#1)$),
        \n1 = {veclen(\p{OA})},
        \n2 = {veclen(\p{OP})},
        \n3 = {scalar(\n1/\n2)}
      in ($(#1)!\n3!{acos(\n1/\n2)}:(#3)$)
    }
  },
  % external homothetic center
  % O1,A1: center of circle 1 and an abitary point on
  ↪ the circle
  % O2,A2: center of circle 2 and an abitary point on
  ↪ the circle
  external center/.style args = {#1,#2,#3,#4}{
    insert path = {
      let
        \p{O1A1} = ($(#2)-(#1)$),% 半径 O1A1
        \p{O2A2} = ($(#4)-(#3)$),% 半径 O2A2
        \n{r1} = {veclen(\p{O1A1})},
        \n{r2} = {veclen(\p{O2A2})},
        \n1 = {scalar(\n{r1}/(\n{r1}-\n{r2}))}
      in ($(#1)!\n1!(#3)$)
    }
  },
  % internal homothetic center
  % O1,A1: center of circle 1 and an abitary point on
  ↪ the circle
  % O2,A2: center of circle 2 and an abitary point on
  ↪ the circle
  internal center/.style args = {#1,#2,#3,#4}{
    insert path = {
```

```
      let
        \p{O1A1} = ($(#2)-(#1)$),% 半径 O1A1
        \p{O2A2} = ($(#4)-(#3)$),% 半径 O2A2
        \n{r1} = {veclen(\p{O1A1})},
        \n{r2} = {veclen(\p{O2A2})},
        \n1 = {scalar(\n{r1}/(\n{r1}+\n{r2}))}
      in ($(#1)!\n1!(#3)$)
    }
  },
  % creates the radical axis of two non-concentric
  ↪ circles
  % O1,A1: center of circle 1 and an abitary point on
  ↪ the circle
  % O2,A2: center of circle 2 and an abitary point on
  ↪ the circle
  radical axis/.style args = {#1,#2,#3,#4}{
    insert path = {
      let
        \n{s} = {\pgfkeysvalueof{/tikz/start modifier}},
        \n{e} = {\pgfkeysvalueof{/tikz/end modifier}},
        \p{O1A1} = ($(#2)-(#1)$),% 半径 O1A1
        \p{O2A2} = ($(#4)-(#3)$),% 半径 O2A2
        \p{O1O2} = ($(#3)-(#1)$),
        \n{r1} = {veclen(\p{O1A1})},
        \n{r2} = {veclen(\p{O2A2})},
        \n{d} = {veclen(\p{O1O2})},
        \n1 = {scalar(\n{r1}/\n{d})},
        \n2 = {scalar(\n{r2}/\n{d})},
        \n3 = {.5*(1+\n1*\n1-\n2*\n2)},
        \p{ft} = ($(#1)!\n3!(#3)$),% perpendicular foot
```

```
        \p{s0} = ($(\p{ft})+(-\y{O1O2},\x{O1O2})$),
        \p{e0} = ($(\p{ft})+(\y{O1O2},-\x{O1O2})$),
        \p{s} = ($(\p{s0})!\n{s}!(\p{e0})$),% start
        \p{e} = ($(\p{s0})!\n{e}!(\p{e0})$)% end
      in (\p{s}) -- (\p{e})
    }
  },
  % circle-line instersections (named: cl1, cl2) of
  ↪ circle (O, A)
  % and line (P, Q)
  % 1. find solutions to equations:
  %    x^2 + y^2 = ra^2 and x = a
  %    wherein a = the (signed) distance from circle
  ↪ center to line
  % 2. rotate and then shift the intersections to get
  ↪ the
  %    solutions to original coordinates
  circle-line/.code args = {#1,#2,#3,#4}{%
    \newdimen\xO
    \newdimen\yO
    \newdimen\xA
    \newdimen\yA
    \newdimen\xP
    \newdimen\yP
    \newdimen\xQ
    \newdimen\yQ
    \pgfextractx{\xO}{\pgfpointanchor{#1}{center}}
    \pgfextracty{\yO}{\pgfpointanchor{#1}{center}}
    \pgfextractx{\xA}{\pgfpointanchor{#2}{center}}
    \pgfextracty{\yA}{\pgfpointanchor{#2}{center}}
```

```
\pgfextractx{\xP}{\pgfpointanchor{#3}{center}}
\pgfextracty{\yP}{\pgfpointanchor{#3}{center}}
\pgfextractx{\xQ}{\pgfpointanchor{#4}{center}}
\pgfextracty{\yQ}{\pgfpointanchor{#4}{center}}
\pgfmathsetmacro{\xa}{\xO / 1cm}
\pgfmathsetmacro{\ya}{\yO / 1cm}
\pgfmathsetmacro{\xp}{\xP / 1cm}
\pgfmathsetmacro{\yp}{\yP / 1cm}
\pgfmathsetmacro{\xq}{\xQ / 1cm}
\pgfmathsetmacro{\yq}{\yQ / 1cm}
\pgfmathsetmacro{\ra}{veclen(\xA-\xO, \yA-\yO) /
↪  1cm}
% 考虑向量 (xq-xp, yq-yp) 和 (x-xp, y-yp) 的叉乘
% C = (xq-xp)*(y-yp)-(x-xp)(yq-yp)
% C = 0: (x,y) 在 PQ 直线上
% C > 0: (x,y) 在 PQ 左侧
% C < 0: (x,y) 在 PQ 右侧
% 设 C = u*x+v*y+w, 则:
\pgfmathsetmacro{\u}{-(\yq-\yp)}
\pgfmathsetmacro{\v}{\xq-\xp}
\pgfmathsetmacro{\w}{\xp*(\yq-\yp)-\yp*(\xq-\xp)}

\pgfmathsetmacro{\temp}{sqrt((\u)^2+(\v)^2)}%\u \v
↪  可能为-, 需加括号
% 计算变换坐标系后的交点坐标 (a,b), 此时 PQ 方向是 y
↪  轴正方向
% 计算圆与直线之间的距离 a, a > 0 时, 圆心在 PQ 左侧;
↪  a < 0 时, 圆心在 PQ 右侧
\pgfmathsetmacro{\a}{(\u*\xa+\v*\ya+\w)/\temp}
% 计算垂直偏移 b (PQ 方向为正方向)
```

```
    \pgfmathsetmacro{\b}{sqrt(\ra^2-(\a)^2)}% 注意加括号
    % 将 (\a, \b) 逆旋转和逆平移变换，得到原坐标系的坐标
    % 计算单位向量 (PQ 方向顺时针 90):
    \pgfmathsetmacro{\c}{-\u/\temp}% cos theta
    \pgfmathsetmacro{\s}{-\v/\temp}% sin theta

    \typeout{========================}
    \typeout{Circle: center (\xa, \ya) radius \ra}
    \typeout{Line coefficients: \u, \v, \w}
    \typeout{Circle-line intersection(simplified): a:\a,
    ↪  b:\b}
    % 旋转矩阵 [c -s; s c]
    \typeout{theta: [cos(theta) sin(theta): \c, \s]}
    \typeout{========================}

    % 两圆交点的坐标
    \coordinate (cl1) at
    ↪  (\xa+\a*\c-\b*\s,\ya+\a*\s+\b*\c);
    \coordinate (cl2) at
    ↪  (\xa+\a*\c+\b*\s,\ya+\a*\s-\b*\c);
  },
  % circle-circle instersections (named: cc1, cc2) of
  ↪  circle
  % O1(A, B) and circle O2(C, D)
  % 1. find solutions to equations:
  %    x^2 + y^2 = ra^2 and (x-d)^2 + y^2 = rb^2
  %    wherein d = the distance of two centers
  % 2. rotate and then shift the intersections to get
  ↪  the
  %    solutions to original coordinates
```

```
circle-circle/.code args = {#1,#2,#3,#4}{%
  \newdimen\xA% center 1
  \newdimen\yA
  \newdimen\xB
  \newdimen\yB
  \newdimen\xC% center 2
  \newdimen\yC
  \newdimen\xD
  \newdimen\yD
  \pgfextractx{\xA}{\pgfpointanchor{#1}{center}}
  \pgfextracty{\yA}{\pgfpointanchor{#1}{center}}
  \pgfextractx{\xB}{\pgfpointanchor{#2}{center}}
  \pgfextracty{\yB}{\pgfpointanchor{#2}{center}}
  \pgfextractx{\xC}{\pgfpointanchor{#3}{center}}
  \pgfextracty{\yC}{\pgfpointanchor{#3}{center}}
  \pgfextractx{\xD}{\pgfpointanchor{#4}{center}}
  \pgfextracty{\yD}{\pgfpointanchor{#4}{center}}
  \pgfmathsetmacro{\xa}{\xA / 1cm}
  \pgfmathsetmacro{\ya}{\yA / 1cm}
  \pgfmathsetmacro{\ra}{veclen(\xB-\xA, \yB-\yA) /
  ↪  1cm}
  \pgfmathsetmacro{\xb}{\xC / 1cm}
  \pgfmathsetmacro{\yb}{\yC / 1cm}
  \pgfmathsetmacro{\rb}{veclen(\xD-\xC, \yD-\yC) /
  ↪  1cm}
  % 计算圆心之间的距离 d
  \pgfmathsetmacro{\d}{sqrt((\xb-\xa)^2+(\yb-\ya)^2)}
  % 计算中间值 a（沿圆心连线方向的距离），可能为 -
  \pgfmathsetmacro{\a}{(\ra^2-\rb^2+\d^2)/(2*\d)}
  % 计算垂直偏移 b(交点到圆心连线的垂直距离)
```

```
% 注意 : \a 可能为-，应加括号
\pgfmathsetmacro{\b}{sqrt(\ra^2-(\a)^2)}

% 将 (\a, \b) 逆旋转和逆平移变换，得到原坐标系的坐标
% 计算单位向量（从第一圆心指向第二圆心的方向）:
\pgfmathsetmacro{\c}{(\xb-\xa)/\d}% cos theta
\pgfmathsetmacro{\s}{(\yb-\ya)/\d}% sin theta

\typeout{========================}
\typeout{Distance: \d}
\typeout{Circle-circle intersection(simplified):
↪  a:\a, b:\b}
% 旋转矩阵 [c -s; s c]
\typeout{theta: [cos(theta) sin(theta): \c, \s]}
\typeout{========================}

% 两圆交点的坐标
% \pgfkeysalso{/tikz/cc
↪  1={\xa+\a*\c-\b*\s,\ya+\a*\s+\b*\c}}%
% \pgfkeysalso{/tikz/cc
↪  2={\xa+\a*\c+\b*\s,\ya+\a*\s-\b*\c}}%
% 可以显示只是算式，未算出结果 在 coordinate 中再计算
↪  出
% \typeout{Point 1:
↪  (\xa+\a*\c-\b*\s,\ya+\a*\s+\b*\c)}
% \typeout{Point 2:
↪  (\xa+\a*\c+\b*\s,\ya+\a*\s-\b*\c)}
\coordinate (cc1) at
↪  (\xa+\a*\c-\b*\s,\ya+\a*\s+\b*\c);
```

```
    \coordinate (cc2) at
    ↪ (\xa+\a*\c+\b*\s,\ya+\a*\s-\b*\c);
  },
  % tangent lines to circle: circle-tagent = {O,A,P},
  ↪ create
  % tangent points (named: tp1, tp2, tp1 is on the left
  ↪ side of OP )
  % O,A: center of circle and an abitary point on the
  ↪ circle
  % P: a point outside the circle
  circle-tangent/.code args = {#1,#2,#3} {
    \coordinate (euclidea@mid) at ($(#1)!.5!(#3)$);
    \tikzset{circle-circle={#1,#2,euclidea@mid,#1},}
    \coordinate (tp1) at (cc1);
    \coordinate (tp2) at (cc2);
  },
  % ========== Path Definitions ==========
  % perpendicular bisector of the line segment (#1 --
  ↪ #2)
  perpendicular bisector/.style args = {#1,#2}{
    insert path = {
      let
        \n{s} = {\pgfkeysvalueof{/tikz/start modifier}},
        \n{e} = {\pgfkeysvalueof{/tikz/end modifier}},
        \p{AB} = ($(#2)-(#1)$),
        \p{m} = ($(#1)!0.5!(#2)$),% midpoint
        \p{s0} = ($(\p{m})+(-\y{AB},\x{AB})$),% rotate
        ↪ ccw, default start
        \p{e0} = ($(\p{m})+(\y{AB},-\x{AB})$),% rotate
        ↪ cw, default end
```

```
      \p{s} = ($(\p{s0})!\n{s}!(\p{e0})$),% start
      \p{e} = ($(\p{s0})!\n{e}!(\p{e0})$)% end
    in (\p{s}) -- (\p{e})
  }
},
% perpendicular line of the line (#1 -- #2) through #3
% specifying start and end with modifiers(see tikz
↪ manual 13.5)
perpendicular/.style args = {#1,#2,#3}{
  insert path = {
    let
      \n{s} = {\pgfkeysvalueof{/tikz/start modifier}},
      \n{e} = {\pgfkeysvalueof{/tikz/end modifier}},
      \p{AB} = ($(#2)-(#1)$),
      \p{ft} = ($(#1)!(#3)!(#2)$),% perpedicular foot
      ↪
      \p{s0} = ($(\p{ft})+(-\y{AB},\x{AB})$),
      \p{e0} = ($(\p{ft})+(\y{AB},-\x{AB})$),
      \p{s} = ($(\p{s0})!\n{s}!(\p{e0})$),% start
      \p{e} = ($(\p{s0})!\n{e}!(\p{e0})$)% end
    in (\p{s}) -- (\p{e})
  }
},
% parallel line of the line (#1 -- #2) through #3
% specifying start and end with modifiers(see tikz
↪ manual 13.5)
parallel/.style args = {#1,#2,#3}{
  insert path = {
    let
      \n{s} = {\pgfkeysvalueof{/tikz/start modifier}},
```

```
        \n{e} = {\pgfkeysvalueof{/tikz/end modifier}},
        \p{s0} = (#3),
        \p{e0} = ($(#3)+(#2)-(#1)$),
        \p{s} = ($(\p{s0})!\n{s}!(\p{e0})$),% start
        \p{e} = ($(\p{s0})!\n{e}!(\p{e0})$)% end
      in (\p{s}) -- (\p{e})
    }
},
% alias for parallel={A,B,A}
extend/.style args = {#1,#2} {
  parallel={#1,#2,#1}
},
% rotate around the origin by `angle` and then
% shift by (`xshift`,`yshift`)
% tansform = {angle:(xshift,yshift)}
transform/.style args = {#1:(#2,#3)} {
  cm={cos(#1), sin(#1), -sin(#1), cos(#1), (#2,#3)}
},
% ========== Conics Definitions ==========
% ellipse with foci and a point
ellipse/a/.initial = 0,
ellipse/b/.initial = 0,
ellipse/c/.initial = 0,
ellipse/e/.initial = 0,
ellipse/angle/.initial = 0,
ellipse/xcenter/.initial = 0,
ellipse/ycenter/.initial = 0,
ellipse/define/.code args = {#1,#2,#3}{
  % 提取各点的坐标分量 x-coordinate and y-coordinate
  % 注意：根据 LaTeX 的规则，宏不能含有数字，
```

```
% 提取坐标的 x 和 y 分量的宏不要含有数字
\newdimen\xFL
\newdimen\yFL
\newdimen\xFR
\newdimen\yFR
\newdimen\xP
\newdimen\yP
\pgfextractx{\xFL}{\pgfpointanchor{#1}{center}}
\pgfextracty{\yFL}{\pgfpointanchor{#1}{center}}
\pgfextractx{\xFR}{\pgfpointanchor{#2}{center}}
\pgfextracty{\yFR}{\pgfpointanchor{#2}{center}}
\pgfextractx{\xP}{\pgfpointanchor{#3}{center}}
\pgfextracty{\yP}{\pgfpointanchor{#3}{center}}

% semimajor axis: a = |PF1 - PF2| / 2
\pgfmathsetmacro{\PFLeft}{veclen(\xP-\xFL,
↪  \yP-\yFL)} % 计算 PF1
\pgfmathsetmacro{\PFRight}{veclen(\xP-\xFR,
↪  \yP-\yFR)} % 计算 PF2
\pgfmathsetmacro{\a}{abs(\PFLeft + \PFRight)/2cm} %
↪  计算 a 并转换单位

% the distance from the center to a focus: c =
↪  |F1F2| / 2
\pgfmathsetmacro{\c}{veclen(\xFL-\xFR, \yFL-\yFR) /
↪  2cm}

% semiminor axis: b = sqrt(a^2 - c^2)
\pgfmathsetmacro{\b}{sqrt(\a^2 - \c^2)}
```

```
% ellipse eccentricity
\pgfmathsetmacro{\e}{\c/\a}

% ellipse center
\pgfmathsetmacro{\xcenter}{(\xFL + \xFR) / 2cm}
\pgfmathsetmacro{\ycenter}{(\yFL + \yFR) / 2cm}

% angle of rotation
\pgfmathsetmacro{\angle}{atan2(\yFR-\yFL,
↪  \xFR-\xFL)}

\pgfkeysalso{/tikz/ellipse/a = \a}
\pgfkeysalso{/tikz/ellipse/b = \b}
\pgfkeysalso{/tikz/ellipse/c = \c}
\pgfkeysalso{/tikz/ellipse/e = \e}
\pgfkeysalso{/tikz/ellipse/xcenter = \xcenter}
\pgfkeysalso{/tikz/ellipse/ycenter = \ycenter}
\pgfkeysalso{/tikz/ellipse/angle = \angle}
\typeout{========================}
\typeout{
  ellipse:{
    a:\pgfkeysvalueof{/tikz/ellipse/a},
    b:\pgfkeysvalueof{/tikz/ellipse/b},
    c:\pgfkeysvalueof{/tikz/ellipse/c},
    angle:\pgfkeysvalueof{/tikz/ellipse/angle},
    center:(\pgfkeysvalueof{/tikz/ellipse/xcenter},
      \pgfkeysvalueof{/tikz/ellipse/ycenter})
  }
}
\typeout{========================}
```

```
    },
    ellipse/.style = {
      insert path = {
        let
          \n{a} = {\pgfkeysvalueof{/tikz/ellipse/a}},
          \n{b} = {\pgfkeysvalueof{/tikz/ellipse/b}},
          \n{angle} =
          ↪ {\pgfkeysvalueof{/tikz/ellipse/angle}},
          \n{xcenter} =
          ↪ {\pgfkeysvalueof{/tikz/ellipse/xcenter}},
          \n{ycenter} =
          ↪ {\pgfkeysvalueof{/tikz/ellipse/ycenter}},
        in [transform={\n{angle}:(\n{xcenter},\n{ycenter}」
        ↪ )}]
          (0,0) ellipse [x radius=\n{a},y radius=\n{b}]
      }
    },
    % ellipse directrices: k = 1.25
    %   (-a^2/c, -k*b) -- (-a^2/c, k*b)
    %   ( a^2/c, -k*b) -- ( a^2/c, k*b)
    ellipse/directrix/scale/.initial = 1.25,
    % ellipse/directrix/scale/.code args= {#1}{
    %   \pgfkeysalso{/tikz/ellipse/directrix/scale = #1}
    % },
    ellipse/directrix/.style = {
      insert path = {
        let
          \n{a} = {\pgfkeysvalueof{/tikz/ellipse/a}},
          \n{b} = {\pgfkeysvalueof{/tikz/ellipse/b}},
          \n{c} = {\pgfkeysvalueof{/tikz/ellipse/c}},
```

```
        \n{d} = {\n{a}^2/\n{c}},
        \n{scale} = {\pgfkeysvalueof{/tikz/ellipse/
        ↪ directrix/scale}},
        \n{angle} =
        ↪ {\pgfkeysvalueof{/tikz/ellipse/angle}},
        \n{xcenter} =
        ↪ {\pgfkeysvalueof{/tikz/ellipse/xcenter}},
        \n{ycenter} =
        ↪ {\pgfkeysvalueof{/tikz/ellipse/ycenter}},
      in [transform={\n{angle}:(\n{xcenter},\n{ycenter}
      ↪ )}]
        (-\n{d}, -\n{b}*\n{scale}) -- (-\n{d},
        ↪ \n{b}*\n{scale})
        ( \n{d}, -\n{b}*\n{scale}) -- ( \n{d},
        ↪ \n{b}*\n{scale})
    }
  },
  % ellipse axes: k = 1.5
  %   (-k*a, 0) -- (k*a, 0)
  %   (0, -k*b) -- (0, k*b)
  ellipse/axis/scale/.initial = 1.5,
  % ellipse/axis/scale/.code args= {#1}{
  %   \pgfkeysalso{/tikz/ellipse/axis/scale = #1}
  % },
  ellipse/axis/.style = {
    insert path = {
      let
        \n{a} = {\pgfkeysvalueof{/tikz/ellipse/a}},
        \n{b} = {\pgfkeysvalueof{/tikz/ellipse/b}},
```

```
        \n{scale} =
         ↪ {\pgfkeysvalueof{/tikz/ellipse/axis/scale}},
        \n{angle} =
         ↪ {\pgfkeysvalueof{/tikz/ellipse/angle}},
        \n{xcenter} =
         ↪ {\pgfkeysvalueof{/tikz/ellipse/xcenter}},
        \n{ycenter} =
         ↪ {\pgfkeysvalueof{/tikz/ellipse/ycenter}},
      in [transform={\n{angle}:(\n{xcenter},\n{ycenter}」
       ↪ )}]
        (-\n{scale}*\n{a}, 0) -- (\n{scale}*\n{a}, 0)
        (0, -\n{scale}*\n{b}) -- (0, \n{scale}*\n{b})
    }
  },
  % hyperbola with foci and a point
  % hyperbola parametric equation: x = a*cosh(t), y =
   ↪ b*sinh(t)
  hyperbola/domain/.initial=-1.5:1.5,
  % hyperbola/doamin/.code args = {#1}{
  %   \pgfkeysalso{/tikz/hyperbola/domain = #1}
  % },
  hyperbola/a/.initial = 0,
  hyperbola/b/.initial = 0,
  hyperbola/c/.initial = 0,
  hyperbola/e/.initial = 0,
  hyperbola/angle/.initial = 0,
  hyperbola/xcenter/.initial = 0,
  hyperbola/ycenter/.initial = 0,
  hyperbola/define/.code args = {#1,#2,#3}{
    % 提取各点的坐标分量 x-coordinate and y-coordinate
```

```
% 注意：根据 LaTeX 的规则，宏不能含有数字，
% 提取坐标的 x 和 y 分量的宏不要含有数字
\newdimen\xFL
\newdimen\yFL
\newdimen\xFR
\newdimen\yFR
\newdimen\xP
\newdimen\yP
\pgfextractx{\xFL}{\pgfpointanchor{#1}{center}}
\pgfextracty{\yFL}{\pgfpointanchor{#1}{center}}
\pgfextractx{\xFR}{\pgfpointanchor{#2}{center}}
\pgfextracty{\yFR}{\pgfpointanchor{#2}{center}}
\pgfextractx{\xP}{\pgfpointanchor{#3}{center}}
\pgfextracty{\yP}{\pgfpointanchor{#3}{center}}

% semimajor axis: a = |PF1 - PF2| / 2
\pgfmathsetmacro{\PFLeft}{veclen(\xP-\xFL,
↪  \yP-\yFL)} % 计算 PF1
\pgfmathsetmacro{\PFRight}{veclen(\xP-\xFR,
↪  \yP-\yFR)} % 计算 PF2
\pgfmathsetmacro{\a}{abs(\PFLeft - \PFRight)/2cm} %
↪  计算 a 并转换单位

% the distance from the center to a focus: c =
↪  |F1F2| / 2
\pgfmathsetmacro{\c}{veclen(\xFL-\xFR, \yFL-\yFR) /
↪  2cm}

% semiminor axis: b = sqrt(c^2 - a^2)
\pgfmathsetmacro{\b}{sqrt(\c^2 - \a^2)}
```

```
% hyperbola eccentricity
\pgfmathsetmacro{\e}{\c/\a}

% hyperbola center
\pgfmathsetmacro{\xcenter}{(\xFL + \xFR) / 2cm}
\pgfmathsetmacro{\ycenter}{(\yFL + \yFR) / 2cm}

% angle of rotation
\pgfmathsetmacro{\angle}{atan2(\yFR-\yFL,
↪  \xFR-\xFL)}

\pgfkeysalso{/tikz/hyperbola/a = \a}
\pgfkeysalso{/tikz/hyperbola/b = \b}
\pgfkeysalso{/tikz/hyperbola/c = \c}
\pgfkeysalso{/tikz/hyperbola/e = \e}
\pgfkeysalso{/tikz/hyperbola/xcenter = \xcenter}
\pgfkeysalso{/tikz/hyperbola/ycenter = \ycenter}
\pgfkeysalso{/tikz/hyperbola/angle = \angle}
\typeout{========================}
\typeout{
  hyperbola:{
    a:\pgfkeysvalueof{/tikz/hyperbola/a},
    b:\pgfkeysvalueof{/tikz/hyperbola/b},
    c:\pgfkeysvalueof{/tikz/hyperbola/c},
    angle:\pgfkeysvalueof{/tikz/hyperbola/angle},
    center:(\pgfkeysvalueof{/tikz/hyperbola/
    ↪  xcenter},
      \pgfkeysvalueof{/tikz/hyperbola/ycenter})
  }
```

```
    }
    \typeout{========================}
  },
  hyperbola/.style = {
    insert path = {
      let
        \n{a} = {\pgfkeysvalueof{/tikz/hyperbola/a}},
        \n{b} = {\pgfkeysvalueof{/tikz/hyperbola/b}},
        \n{angle} =
        ↪ {\pgfkeysvalueof{/tikz/hyperbola/angle}},
        \n{xcenter} =
        ↪ {\pgfkeysvalueof{/tikz/hyperbola/xcenter}},
        \n{ycenter} =
        ↪ {\pgfkeysvalueof{/tikz/hyperbola/ycenter}},
      in [smooth,domain=\pgfkeysvalueof{/tikz/
      ↪ hyperbola/domain},
      variable=\euclidea@var,transform={\n{angle}:(\n{
      ↪ xcenter},\n{ycenter})}]
        plot({-\n{a}*cosh(\euclidea@var)},{\n{b}*sinh(\
        ↪ euclidea@var)})  % right branch
        plot({ \n{a}*cosh(\euclidea@var)},{\n{b}*sinh(\
        ↪ euclidea@var)})  % left branch
    }
  },
  % asymptote parametric equation: x = a*sinh(t), y =
  ↪ b*sinh(t)
  asymptote/.style = {
    insert path = {
      let
        \n{a} = {\pgfkeysvalueof{/tikz/hyperbola/a}},
```

```
        \n{b} = {\pgfkeysvalueof{/tikz/hyperbola/b}},
        \n{angle} =
        ↪ {\pgfkeysvalueof{/tikz/hyperbola/angle}},
        \n{xcenter} =
        ↪ {\pgfkeysvalueof{/tikz/hyperbola/xcenter}},
        \n{ycenter} =
        ↪ {\pgfkeysvalueof{/tikz/hyperbola/ycenter}},
      in [smooth,domain=\pgfkeysvalueof{/tikz/
      ↪ hyperbola/domain},
      variable=\euclidea@var,transform={\n{angle}:(\n{
      ↪ xcenter},\n{ycenter})}]
        plot({-\n{a}*sinh(\euclidea@var)},{\n{b}*sinh(\
        ↪ euclidea@var)})
        plot({ \n{a}*sinh(\euclidea@var)},{\n{b}*sinh(\
        ↪ euclidea@var)})
    }
  },
  % hyperbola directrices: k = 1.5
  %   (-a^2/c, -k*b) -- (-a^2/c, k*b)
  %   ( a^2/c, -k*b) -- ( a^2/c, k*b)
  hyperbola/directrix/scale/.initial = 1.5,
  % hyperbola/directrix/scale/.code args= {#1}{
  %   \pgfkeysalso{/tikz/hyperbola/directrix/scale = #
1}
  % },
  hyperbola/directrix/.style = {
    insert path = {
      let
        \n{a} = {\pgfkeysvalueof{/tikz/hyperbola/a}},
        \n{b} = {\pgfkeysvalueof{/tikz/hyperbola/b}},
```

```
        \n{c} = {\pgfkeysvalueof{/tikz/hyperbola/c}},
        \n{d} = {\n{a}^2/\n{c}},
        \n{scale} = {\pgfkeysvalueof{/tikz/hyperbola/
        ↪ directrix/scale}},
        \n{angle} =
        ↪ {\pgfkeysvalueof{/tikz/hyperbola/angle}},
        \n{xcenter} =
        ↪ {\pgfkeysvalueof{/tikz/hyperbola/xcenter}},
        \n{ycenter} =
        ↪ {\pgfkeysvalueof{/tikz/hyperbola/ycenter}},
      in [transform={\n{angle}:(\n{xcenter},\n{ycenter}
      ↪ )}]
        (-\n{d}, -\n{scale}*\n{b}) -- (-\n{d},
        ↪ \n{scale}*\n{b})
        ( \n{d}, -\n{scale}*\n{b}) -- ( \n{d},
        ↪ \n{scale}*\n{b})
    }
  },
  % hyperbola axes: k = 1.5
  %   (-k*c, 0) -- (k*c, 0)
  %   (0, -k*b) -- (0, k*b)
  hyperbola/axis/scale/.initial = 1.5,
  % hyperbola/axis/scale/.code args= {#1}{
  %   \pgfkeysalso{/tikz/hyperbola/axis/scale = #1}
  % },
  hyperbola/axis/.style = {
    insert path = {
      let
        \n{b} = {\pgfkeysvalueof{/tikz/hyperbola/b}},
        \n{c} = {\pgfkeysvalueof{/tikz/hyperbola/c}},
```

```
        \n{scale} = {\pgfkeysvalueof{/tikz/hyperbola/
        ↪ axis/scale}},
        \n{angle} =
        ↪ {\pgfkeysvalueof{/tikz/hyperbola/angle}},
        \n{xcenter} =
        ↪ {\pgfkeysvalueof{/tikz/hyperbola/xcenter}},
        \n{ycenter} =
        ↪ {\pgfkeysvalueof{/tikz/hyperbola/ycenter}},
      in [transform={\n{angle}:(\n{xcenter},\n{ycenter}
      ↪ )}]
        (-\n{scale}*\n{c}, 0) -- (\n{scale}*\n{c}, 0)
        (0, -\n{scale}*\n{b}) -- (0, \n{scale}*\n{b})
    }
  },
  % parabola with the focus and vertex
  % parabola parametric equation:
  % y^2 = 4a*x, x = a*t^2, y= 2a*t
  % The quantity 4a is known as the latus rectum.
  parabola/domain/.initial=-2:2,
  % parabola/doamin/.code args = {#1}{
  %   \pgfkeysalso{/tikz/parabola/domain = #1}
  % },
  parabola/a/.initial = 0,
  parabola/e/.initial = 1,
  parabola/angle/.initial = 0,
  parabola/@xvertex/.initial = 0,
  parabola/@yvertex/.initial = 0,
  parabola/define/.code args = {#1,#2}{
    % 提取各点的坐标分量 x-coordinate and y-coordinate
    % 注意：根据 LaTeX 的规则，宏不能含有数字，
```

```
% 提取坐标的 x 和 y 分量的宏不要含有数字
\newdimen\xF
\newdimen\yF
\newdimen\xV
\newdimen\yV
\pgfextractx{\xF}{\pgfpointanchor{#1}{center}}
\pgfextracty{\yF}{\pgfpointanchor{#1}{center}}
\pgfextractx{\xV}{\pgfpointanchor{#2}{center}}
\pgfextracty{\yV}{\pgfpointanchor{#2}{center}}

% latus rectum 4a
% 注意: semi-latus rectum 通常表示为 p (=2a),
% macro 不要定义为 \p, \p 有表示向量的含义
\pgfmathsetmacro{\a}{veclen(\xV-\xF, \yV-\yF)/1cm}
↪  % 计算 PF

% parabola vertex
\pgfmathsetmacro{\xvertex}{(\xV) / 1cm}
\pgfmathsetmacro{\yvertex}{(\yV) / 1cm}

% angle of rotation
\pgfmathsetmacro{\angle}{atan2(\yF-\yV, \xF-\xV)}

\pgfkeysalso{/tikz/parabola/a = \a}
\pgfkeysalso{/tikz/parabola/@xvertex = \xvertex}
\pgfkeysalso{/tikz/parabola/@yvertex = \yvertex}
\pgfkeysalso{/tikz/parabola/angle = \angle}
\typeout{=========================}
\typeout{
  parabola:{
```

```
        a:\pgfkeysvalueof{/tikz/parabola/a},
        angle:\pgfkeysvalueof{/tikz/parabola/angle},
        center:(\pgfkeysvalueof{/tikz/parabola/
        ↪ @xvertex},
          \pgfkeysvalueof{/tikz/parabola/@yvertex})
      }
    }
    \typeout{========================}
  },
  parabola/.style = {
    insert path = {
      let
        \n{a} = {\pgfkeysvalueof{/tikz/parabola/a}},
        \n{angle} =
        ↪ {\pgfkeysvalueof{/tikz/parabola/angle}},
        \n{xvertex} =
        ↪ {\pgfkeysvalueof{/tikz/parabola/@xvertex}},
        \n{yvertex} =
        ↪ {\pgfkeysvalueof{/tikz/parabola/@yvertex}},
      in [smooth,domain=\pgfkeysvalueof{/tikz/parabola/
      ↪ domain},
      variable=\euclidea@var,
      transform={\n{angle}:(\n{xvertex},\n{yvertex})}]
        plot({\n{a}*(\euclidea@var)^2},{2*\n{a}*(\
        ↪ euclidea@var)})
    }
  },
  % parabola directrix: k = 2
  %   (-a, -k*a) -- (-a, 2k*a)
  parabola/directrix/scale/.initial = 2,
```

```
% parabola/directrix/scale/.code args= {#1}{
%   \pgfkeysalso{/tikz/parabola/directrix/scale = #1}
% },
parabola/directrix/.style = {
  insert path = {
    let
      \n{a} = {\pgfkeysvalueof{/tikz/parabola/a}},
      \n{scale} = {\pgfkeysvalueof{/tikz/parabola/
      ↪ directrix/scale}},
      \n{angle} =
      ↪ {\pgfkeysvalueof{/tikz/parabola/angle}},
      \n{xcenter} =
      ↪ {\pgfkeysvalueof{/tikz/parabola/@xvertex}},
      \n{ycenter} =
      ↪ {\pgfkeysvalueof{/tikz/parabola/@yvertex}},
    in [transform={\n{angle}:(\n{xcenter},\n{ycenter}
    ↪ )}]
      (-\n{a}, -2*\n{scale}*\n{a}) -- (-\n{a},
      ↪ 2*\n{scale}*\n{a})
  }
},
% parabola directrix: k = 2
%   (-2a, 0) -- (k^2*a, 0)
parabola/axis/scale/.initial = 2,
% parabola/axis/scale/.code args= {#1}{
%   \pgfkeysalso{/tikz/parabola/axis/scale = #1}
% },
parabola/axis/.style = {
  insert path = {
    let
```

```
        \n{a} = {\pgfkeysvalueof{/tikz/parabola/a}},
        \n{scale} = {\pgfkeysvalueof{/tikz/parabola/
        ↪ axis/scale}},
        \n{angle} =
        ↪ {\pgfkeysvalueof{/tikz/parabola/angle}},
        \n{xcenter} =
        ↪ {\pgfkeysvalueof{/tikz/parabola/@xvertex}},
        \n{ycenter} =
        ↪ {\pgfkeysvalueof{/tikz/parabola/@yvertex}},
      in [transform={\n{angle}:(\n{xcenter},\n{ycenter}
      ↪ )}]
        (-2*\n{a}, 0) -- (\n{scale}^2*\n{a},0)
    }
  },
  conic/define/.code args = {#1,#2,#3,#4,#5,#6}{
    \tikzset {
      conic/matrix={\cs@Q,#1,#2,#3,#4,#5,#6},
      conic/reduce={\cs@Q,\cs@R,\cs@angle,\cs@xshift,\
      ↪ cs@yshift},
    }
    \tikzmath {
      % \xshift = - \xshift;
      % \yshift = - \yshift;
      real \cs@a, \cs@b;
      \cs@a = 0.0; \cs@b = 0.0;
      if \cs@R{1,1} > \EPSILON  && \cs@R{2,2} > \EPSILON
      ↪ then {%ellipse
        \cs@type = 1;
        \cs@a = sqrt(1/\cs@R{1,1});
        \cs@b = sqrt(1/\cs@R{2,2});
```

```
    };
    if abs(\cs@R{1,1}) <= \EPSILON then {%parabola:
    ↪  y^2 = 4ax
      \cs@type = 2;
      \cs@a = -\cs@R{1,3}/2;
    };
    if abs(\cs@R{2,2}) <= \EPSILON then {%parabola:
    ↪  x^2 = 4ay
      \cs@type = 2;
      \cs@angle = \cs@angle + 90;
      \cs@a = -\cs@R{2,3}/2;
    };
    if \cs@R{1,1} > \EPSILON  && \cs@R{2,2} <
    ↪  -\EPSILON then {%hyperbola
      \cs@type = 3;
      \cs@a = sqrt( 1/\cs@R{1,1});
      \cs@b = sqrt(-1/\cs@R{2,2});
    };
    if \cs@R{1,1} < -\EPSILON  && \cs@R{2,2} >
    ↪  \EPSILON then {%hyperbola
      \cs@type = 3;
      \cs@angle = \cs@angle + 90;
      \cs@b = sqrt(-1/\cs@R{1,1});
      \cs@a = sqrt( 1/\cs@R{2,2});
    };
  }
  \typeout{========================}
  \typeout{a: \cs@a, b: \cs@b}
  \typeout{type: \cs@type}
```

```
      \typeout{transform:
      ↪ \cs@angle:(\cs@xshift,\cs@yshift)}
      \typeout{=======================}
    },
    conic/.style = {
      insert path = {
        \ifnum \cs@type = 1 %ellipse
          [transform={\cs@angle:(\cs@xshift,\cs@yshift)}]
          (0,0) ellipse [x radius=\cs@a,y radius=\cs@b]
        \else\ifnum \cs@type = 2%parabola
          [smooth,domain=\pgfkeysvalueof{/tikz/parabola/
          ↪ domain},
          variable=\euclidea@var,
          transform={\cs@angle:(\cs@xshift,\cs@yshift)}]
          plot({\cs@a*(\euclidea@var)^2},{2*\cs@a*(\
          ↪ euclidea@var)})
        \else\ifnum \cs@type = 3%hyperbola
          [smooth,domain=\pgfkeysvalueof{/tikz/hyperbola/
          ↪ domain},
          variable=\euclidea@var,transform={\cs@angle:(\
          ↪ cs@xshift,\cs@yshift)}]
          plot({-\cs@a*cosh(\euclidea@var)},{\cs@b*sinh(\
          ↪ euclidea@var)}) % right branch
          plot({ \cs@a*cosh(\euclidea@var)},{\cs@b*sinh(\
          ↪ euclidea@var)}) % left branch
        \fi\fi\fi
      }
    },
  }
```

```
% Utilities for implementation of 'intercept'
\def\euclidea@ComputeLength#1,#2\euclidea@stop{
  \newdimen\euclidea@ax
  \newdimen\euclidea@ay
  \newdimen\euclidea@bx
  \newdimen\euclidea@by
  \pgfextractx{\euclidea@ax}{\pgfpointanchor{#1}{
  ↪ center}}
  \pgfextracty{\euclidea@ay}{\pgfpointanchor{#1}{
  ↪ center}}
  \pgfextractx{\euclidea@bx}{\pgfpointanchor{#2}{
  ↪ center}}
  \pgfextracty{\euclidea@by}{\pgfpointanchor{#2}{
  ↪ center}}
  % 以下 showthe 指令 overleaf.com 编译通过, 而在
  ↪ macOS+texlive 2021 报错
  % \showthe\euclidea@ax
  % \showthe\euclidea@ay
  % \showthe\euclidea@bx
  % \showthe\euclidea@by
  \pgfmathveclen{\euclidea@ax-\euclidea@bx}{\
  ↪ euclidea@ay-\euclidea@by}
}


% Syntax of parabola: y = a * x * x + b * x + c
%   \parabola [options] (a,b,c);
\newcommand\parabola{} % just for safety
\def\parabola[#1]#2(#3,#4,#5){
  \path[smooth,domain=-2:2,#1,variable=\euclidea@var]
```

```
    plot({\euclidea@var},
      {(#3)*\euclidea@var*\euclidea@var+(#4)*\
       ↪  euclidea@var+(#5)})
}

% Syntax of ellipse:
%   \ellipse [options] (a,b);
% wherein, a,b are major/minor semi axis.
\newcommand\ellipse{} % just for safety
\def\ellipse[#1]#2(#3,#4){
  \path[#1] (0,0) ellipse ({#3} and {#4})
}

% Syntax of hyperbola:
%   \hyperbola [options] (a,b);
% wherein, a,b are major/minor semi axis.
\newcommand\hyperbola{} % just for safety
\def\hyperbola[#1]#2(#3,#4){
  \path[smooth,domain=-1.5:1.5,#1,variable=\
   ↪  euclidea@var]
    plot({-(#3)*cosh(\euclidea@var)},
      {(#4)*sinh(\euclidea@var)})% right arm
    plot({ (#3)*cosh(\euclidea@var)},
      {(#4)*sinh(\euclidea@var)})% left arm
}

% Syntax of asymptote:
%   \asymptote [options] (a,b);
% wherein, a,b are major/minor semi axis.
\newcommand\asymptote{} % just for safety
```

```
\def\asymptote[#1]#2(#3,#4){
  \path[domain=-2:2,#1,variable=\euclidea@var,]
    plot({\euclidea@var},{ (#4)/(#3)*(\euclidea@var)})
    plot({\euclidea@var},{-(#4)/(#3)*(\euclidea@var)})
}

% Syntax of axes:
%   \axes (xmin:xmax, ymin:ymax);
\newcommand\axes{}
\def\axes(#1:#2,#3:#4){
  \draw[help lines] (#1,#3) grid[step=1] (#2,#4);
  \draw[-latex] ({(#1)-0.5},0) -- ({(#2)+0.5},0)
  ↪ node[below] {$x$};
  \draw[-latex] (0,{(#3)-0.5}) -- (0,{(#4)+0.5})
  ↪ node[left] {$y$};
  \draw (0,0) node[below left] {$O$}
}

\makeatother
```

D.2 线性代数包

```
\ProvidesFile{tikzlibrarymc.code.tex}[2025/03/19 v1.2.0
↪ A tikz library for matrix computations and linear
↪ algebra]

% 提高数值计算精度
% https://tex.stackexchange.com/questions/521857/tikz-
↪ fpu-seems-to-be-inaccurate
```

```
\RequirePackage{xfp}

\usetikzlibrary{math,calc,quotes}

% ================================================
% Caveats
% ================================================
% 1. Tikzmath is mainly used for handling numerical
 ↪  computations and variable
% assignments, supporting basic mathematical operations
 ↪  and conditional
% statements, but it does not support arrays or lists as
 ↪  function parameters.
% 2. Tikzmath internally requires semicolons for
 ↪  separation, allows comments,
% but cannot have blank lines.
% 3. The for statement does not have loop interruption
 ↪  commands like 'break' or 'continue'.
% 4. The first iteration of a for loop is always
 ↪  executed without evaluating the
% loop variable; to restrict the range of the loop
 ↪  variable, it must be
% explicitly limited.
% 5. Integer "subscripts" (not true subscripts) must be
 ↪  defined as int; otherwise,
% they default to floating-point numbers, and defined
 ↪  macros cannot be found.
% 6. Changes to variables inside a function do not
 ↪  propagate to the outside.
```

```
% 7. When defining functions, there should be no spaces
 ↪ between parameters.
% 8. The return statement does not interrupt the
 ↪ execution of subsequent
% statements within the function.
% 9. The maximum number of function parameters is 9.
% 10.Function parameters cannot directly use arrays.
% 11.Extremely difficult to debug.
% 12.Given the limitations of tikzmath' s function, you
 ↪ can use tikzset
% with handlers(.code) to execute some code.
% 13.Call tikzmath within the code, but note that this
 ↪ may generate
% some global variables; to avoid conflicts with
 ↪ user-defined variables,
% add the prefix mc@.
% 14.Modifications of variables inside a foreach
 ↪ statement will vanish outside.
% 15.Special attention should be paid to variables in
 ↪ tikzmath, as these
% variables are global. If they are used directly
 ↪ without declaration,
% it may pose risks. For example, \mc@c might be
 ↪ declared as an int in one
% place, but if it is assigned a floating-point number
 ↪ elsewhere without
% declaration, it will automatically be converted to an
 ↪ integer.
%
```

```
% 1. tikzmath 主要用于处理数值计算和变量赋值，支持基本的
↪ 数学运算和条件语句，
% 但不支持数组或列表作为函数参数.
% 2. tikzmath 内部要以分号分隔，可以有注释，但不能有空
行.
% 3. for 语句没有类似 break，continue 之类的中断循环的语
↪ 句.
% 4. for 的第一次总是执行的，不对循环变量作判断，要限制循
↪ 环变量的范围.
% 5. 整数" 下标"(非真正下标)，必须定义为 int，否则默认为
↪ 浮点数，无法找到定义的宏.
% 6. 函数内部对变量的修改不会传导至外部.
% 7. 函数定义时，参数之间不要有空格.
% 8. return 语句不会中断函数内部下面语句的执行.
% 9. 函数参数个数最大为 9.
% 10.函数参数不能直接使用数组.
% 11.极难调试.
% 12.鉴于 tikzmath 的 function 存在一些缺陷，可以使用
% tikzset 的 /.code 来执行一些代码
% 13.在代码中调用 tikzmath，但是注意这里会产生一些全局的
↪ 变量，
% 为了避免与用户的变量冲突，添加前缀 mc@
% 14.foreach 中的修改在循环外面是失效的
% https://tex.stackexchange.com/questions/196065/」
↪ pgfkeys-computed-i-e-dynamically-defined-key
% 15.要特别注意 tikzmath 中的变量，这些变量是全局的，如果
↪ 不加声明就直接使用
% 可能会带来风险，如 \mc@c 可能在某处被声明为 int，在另一
↪ 处不声明就赋值浮点数时则会
% 自动转换为整数
```

```
\makeatletter

\tikzmath {
  % threshold, numbers not more than epsilon will be
   ↪ treated as zero.
  \EPSILON = 0.00005;
  % Jacobi maximum number of iterations, default:
   ↪ 3*(n-1)^2
  % wherein, n is size of the symetric matrix,
  % User can override it with a positive integer
  % -1 means using the default value
  \JacobiMaxIter = -1;
  % status of solving linear systems, inversing
   ↪ matrices, and
  % Jacobi eigenvalue algorithm
  % 1 for success and 0 for failure
  \status = 0;
  % Determinant calculation using Gaussian elimination
  % In order to calculate determinant, user should
   ↪ initialize the matrix
  % \mc@mat, and then call this function. This function
   ↪ is reused in
  % calcuate minors of a matrix.
  function det(\n) {
    \result = 1.0;
    int \i, \j, \k;
    int \stop;
    \stop = 0;
    % for 的第一项总是执行的，要限制范围
```

```
    for \k in {1,...,\n-1} {
      if \stop == 0 then {
        \pivotRow = \k;
        \pivot = abs(\mc@mat{\k,\k});
        for \i in {\k+1,...,\n} {
          if \pivot < abs(\mc@mat{\i,\k}) then {
            \pivotRow = \i;
            \pivot = abs(\mc@mat{\i,\k});
          };
        };
        if \pivot < 0.00001 then {
          \stop = 1;
          \result = 0.0;
        };
        if \stop == 0 then {
          % swap
          if \pivotRow != \k then {
            \result = -\result;
            for \j in {\k,...,\n}{
              \temp = \mc@mat{\k,\j};
              \mc@mat{\k,\j} = \mc@mat{\pivotRow,\j};
              \mc@mat{\pivotRow,\j} = \temp;
            };
          };
          % eliminate
          for \i in {\k+1,...,\n}{
            \factor = \fpeval{\mc@mat{\i,\k} /
            ↪  \mc@mat{\k,\k}};
            for \j in {\k,...,\n}{
```

```
            \mc@mat{\i,\j} = \fpeval{\mc@mat{\i,\j} -
            ↪ \factor * \mc@mat{\k,\j}};
          };
        };
        \result = \fpeval{\result * \mc@mat{\k,\k}};
      };% stop
    };% stop
  };
  \result = \fpeval{\result * \mc@mat{\n,\n}};
  return \result;
};
}

\tikzset {
  % display matrix: show={a,m,n}
  show/.code args={#1,#2,#3} {
    \tikzmath{
      print{\ \\\detokenize{#1}\\};% convert to string
      int \mc@m, \mc@n;
      \mc@m = #2;
      \mc@n = #3;
      int \mc@i, \mc@j;
      for \mc@i in {1,...,\mc@m} {
        for \mc@j in {1,...,\mc@n} {
          print{[\mc@i,\mc@j]:\ #1{\mc@i,\mc@j}, \ \ };
        };
        print{\\};
      };
    }%tikzmath
  },
```

```
% scalar multiplication: show={a,m,n,k,b}
% b = k * a
scalar/.code args={#1,#2,#3,#4,#5} {
  \tikzmath{
    int \mc@m, \mc@n;
    \mc@m = #2;
    \mc@n = #3;
    int \mc@i, \mc@j;
    for \mc@i in {1,...,\mc@m} {
      for \mc@j in {1,...,\mc@n} {
        #5{\mc@i,\mc@j} = #4 * #1{\mc@i,\mc@j};
      };
    };
  }%tikzmath
},
% transpose matrix: transpose={\A,\B,\mc@row,\mc@col}
% B = transpose(A)
transpose/.code args={#1,#2,#3,#4} {
  \tikzmath{
    int \mc@m, \mc@n;
    \mc@m = #3;
    \mc@n = #4;
    int \mc@i, \mc@j;
    for \mc@i in {1,...,\mc@n} {
      for \mc@j in {1,...,\mc@m} {
        #2{\mc@i,\mc@j} = #1{\mc@j,\mc@i};
      };
    };
  }%tikzmath
},
```

```
% matrix multiplication:
↪ product={a,b,c,row1,col1,col2}
% returns the matrix product of two matrices: c = a *
↪ b
% size(a) = row1 x col1, size(b) = row2 x col2
% col1 = row2
product/.code args={#1,#2,#3,#4,#5,#6} {
  \tikzmath{
    int \mc@row1, \mc@col1, \mc@col2;
    \mc@row1 = #4;
    \mc@col1 = #5;
    \mc@col2 = #6;
    int \mc@i, \mc@j, \mc@k;
    for \mc@i in {1,...,\mc@row1} {
      for \mc@j in {1,...,\mc@col2} {
        #3{\mc@i,\mc@j} = 0.0;
        for \mc@k in {1,...,\mc@col1} {
          #3{\mc@i,\mc@j} = #3{\mc@i,\mc@j} +
              #1{\mc@i,\mc@k} * #2{\mc@k,\mc@j};
        };
      };
    };
  }%tikzmath
},
% Determinant calculation using Gaussian elimination
↪ with partial pivoting
% det={\a,\n,\det}
% the result is stored in \det,
% the original matrix is not modified.
det/.code args={#1,#2,#3} {
```

```
    \tikzmath{
      int \mc@i, \mc@j, \mc@k, \mc@n;
      \mc@n = #2;
      int \mc@stop;
      \mc@stop = 0;
      % keep the original matrix
      for \mc@i in {1,...,\mc@n}{
        for \mc@j in {1,...,\mc@n}{
          \mc@mat{\mc@i,\mc@j} = #1{\mc@i,\mc@j};
        };
      };
      #3 = det(\mc@n);
    }%tikzmath
  },
  % The (i, j) minor is the det of the submatrix formed
  % by deleting the i-th row and j-th column.
  % The (i, j) cofactor is obtained by multiplying
  % the minor by (-1)^(i + j).
  % minors={\a,\b,\n}
  % the result is stored in \minor,
  % the original matrix is not modified.
  minors/.code args={#1,#2,#3}{
    \tikzmath{
      int \mc@r, \mc@c, \mc@n;
      int \mc@i, \mc@j;
      \mc@n = #3;
      for \mc@r in {1,...,\mc@n}{
        for \mc@c in {1,...,\mc@n}{
          % initalize \mc@mat
          int \mc@nexti, \mc@nextj;
```

```
            for \mc@i in {1,...,\mc@n} {
              for \mc@j in {1,...,\mc@n} {
                if \mc@i != \mc@r && \mc@j != \mc@c then{
                  if \mc@i < \mc@r then {
                    \mc@nexti = \mc@i;
                  } else {
                    \mc@nexti = \mc@i - 1;
                  };
                  if \mc@j < \mc@c then {
                    \mc@nextj = \mc@j;
                  } else {
                    \mc@nextj = \mc@j - 1;
                  };
                  \mc@mat{\mc@nexti,\mc@nextj} =
                  ↪  #1{\mc@i,\mc@j};
                };%if
              };
            };%for
            % calculate det, we cannot directly use
            ↪  \mc@n-1 as bellow
            % #2{\mc@r,\mc@c} = det(\mc@n-1);
            int \mc@dim;
            \mc@dim = \mc@n - 1;
            #2{\mc@r,\mc@c} = det(\mc@dim);
          };
        };
      }% No commas after it
    },
    cofactors/.code args={#1,#2,#3}{
      \tikzset{
```

```
      minors={#1,#2,#3},
    }
    \tikzmath{
      int \mc@r, \mc@c, \mc@n;
      \mc@n = #3;
      for \mc@r in {1,...,\mc@n}{
        for \mc@c in {1,...,\mc@n}{
          #2{\mc@r,\mc@c} = (-1)^(\mc@r+\mc@c) *
          ↪ #2{\mc@r,\mc@c};
        };
      };
    }%tikzmath
  },
  % Solving linear systems using Gaussian-Jordan
  ↪ elimination with partial pivoting
  % solve={a,b,c,row1,col2}
  % solve ax=b, size(a) = row1 x row1, size(b) = row1 x
  ↪ col2, size(c) = row1 x col2
  % if the system is solvable, \status is 1; otherwise
  ↪ 0.
  solve/.code args={#1,#2,#3,#4,#5}{
    \tikzmath{
      % print{Solve linear system...\\};
      \status = 1; % succeed
      int \mc@i, \mc@j, \mc@k;
      int \mc@row1, \mc@col2;
      \mc@row1 = #4;
      \mc@col2 = #5;
      % keep the original matrices
      for \mc@i in {1,...,\mc@row1} {
```

```
    for \mc@j in {1,...,\mc@row1} {
      \mc@a{\mc@i,\mc@j} = #1{\mc@i,\mc@j};
    };
    for \mc@j in {1,...,\mc@col2} {
      #3{\mc@i,\mc@j} = #2{\mc@i,\mc@j};
    };
  };
  int \mc@stop;
  \mc@stop = 0;
  % for 的第一项总是执行的，要限制范围
  for \mc@k in {1,...,\mc@row1} {
    if \mc@stop == 0 then {
      \mc@pivotRow = \mc@k;
      \mc@pivot = abs(\mc@a{\mc@k,\mc@k});
      % pivoting
      if \mc@k < \mc@row1 then {
        for \mc@i in {\mc@k+1,...,\mc@row1} {
          if \mc@pivot < abs(\mc@a{\mc@i,\mc@k})
          ↪ then {
            \mc@pivotRow = \mc@i;
            \mc@pivot = abs(\mc@a{\mc@i,\mc@k});
          };
        };
      };
      if \mc@pivot < \EPSILON then {
        \mc@stop = 1;
        \status = 0; % failed
      };
      if \mc@stop == 0 then {
        % swap
```

```
if \mc@pivotRow != \mc@k then {
  % swap A
  for \mc@j in {\mc@k,...,\mc@row1}{
    \mc@temp = \mc@a{\mc@k,\mc@j};
    \mc@a{\mc@k,\mc@j} =
    ↪  \mc@a{\mc@pivotRow,\mc@j};
    \mc@a{\mc@pivotRow,\mc@j} = \mc@temp;
  };
  % swap B
  for \mc@j in {1,...,\mc@col2}{
    \mc@temp = #3{\mc@k,\mc@j};
    #3{\mc@k,\mc@j} =
    ↪  #3{\mc@pivotRow,\mc@j};
    #3{\mc@pivotRow,\mc@j} = \mc@temp;
  };
};
% eliminate
int \mc@next;
for \mc@next in {0,...,\mc@row1-1}{
  %先处理第 k 行 (i.e. \mc@next=0), 将主元归
  ↪  一化, 然后消去其它行
  \mc@i =int(mod(\mc@k + \mc@next,
  ↪  \mc@row1)) ;
  if \mc@i == 0 then {
    \mc@i = \mc@row1;
  };
  \mc@temp = \mc@a{\mc@i,\mc@k};
  % eliminate A
  for \mc@j in {\mc@k,...,\mc@row1}{
    if \mc@i == \mc@k then {
```

```
                \mc@a{\mc@i,\mc@j} =
                ↪ \fpeval{\mc@a{\mc@i,\mc@j} /
                ↪ \mc@temp};
              } else {%注意不能少 else
                \mc@a{\mc@i,\mc@j} =
                ↪ \fpeval{\mc@a{\mc@i,\mc@j} -
                ↪ \mc@temp * \mc@a{\mc@k,\mc@j}};
              };
            };
            % eliminate B
            for \mc@j in {1,...,\mc@col2}{
              if \mc@i == \mc@k then {
                #3{\mc@i,\mc@j} =
                ↪ \fpeval{#3{\mc@i,\mc@j} /
                ↪ \mc@temp};
              } else {%注意不能少 else
              #3{\mc@i,\mc@j} =
               ↪ \fpeval{#3{\mc@i,\mc@j} - \mc@temp *
               ↪ #3{\mc@k,\mc@j}};
              };
            };
          };
        };% stop
      };% stop
    };%for
  }%tikzmath
},
% inverse matrix: inverse={a,b,n}
% b = inverse(a)
inverse/.code args={#1,#2,#3}{
```

```
    \tikzmath{
      int \mc@n;
      \mc@n = #3;
      % initialize
      for \mc@i in {1,...,\mc@n} {
        for \mc@j in {1,...,\mc@n} {
          if \mc@i == \mc@j then {
            #2{\mc@i,\mc@j} = 1.0;
          } else {
            #2{\mc@i,\mc@j} = 0.0;
          };
        };
      };
    }%tikzmath
    \tikzset {
      solve={#1,#2,#2,#3,#3},
    }
  },
  % Egienvalues and Egienvectors of a symetric matrix
  % jacobi={\a,\d,\v,n}
  % \a: a symetric matrix
  % \d: returns a diagnoalized matrix with eigenvalues
  ↪  on diagonal
  % \v: returns the eigenvector matrix
  % n: size of matrix 'a'
  % change \status to 1 if maximum number of iterations
  % hasn't been exceeded, otherwise 0;
  jacobi/.code args = {#1,#2,#3,#4} {
    \tikzmath{
      \status = 1;
```

```
int \mc@i, \mc@j, \mc@k, \mc@n;
\mc@n = #4;
% max iterations
int \JacobiMaxIter;
if \JacobiMaxIter < 0 then {% not overriden by
↪  user
  \JacobiMaxIter = 3 * (\mc@n-1)^2;
};
% Initialize eigenvector matrix as identity
for \mc@i in {1,...,\mc@n} {
  for \mc@j in {1,...,\mc@n} {
    #2{\mc@i,\mc@j} = #1{\mc@i,\mc@j};
    if \mc@i == \mc@j then {
      #3{\mc@i,\mc@j} = 1.0;
    } else {
      #3{\mc@i,\mc@j} = 0.0;
    };
  };
};
% print{JacobiMaxIter: \JacobiMaxIter\\};
int \mc@stop;
\mc@stop = 0;
for \mc@k in {1,...,\JacobiMaxIter} {
  % Performs a single Jacobi rotation to eliminate
  ↪  an off-diagonal element
  if \mc@stop == 0 then {
    % find the maximum off-diagnoal element
    int \mc@p, \mc@q; % position p < q
    real \mc@temp;
    \mc@p = 1; \mc@q = 2; \mc@temp = 0.0;
```

```
for \mc@i in {1,...,\mc@n-1} {
  for \mc@j in {\mc@i+1,...,\mc@n} {
    % print{[\mc@i,\mc@j]:\
    ↪  #2{\mc@i,\mc@j}\\};
    if abs(#2{\mc@i,\mc@j}) > \mc@temp then {
      \mc@temp = abs(#2{\mc@i,\mc@j});
      \mc@p = \mc@i;
      \mc@q = \mc@j;
    };
  };
};
% print{abs max:[\mc@p,\mc@q]:\
↪  #2{\mc@p,\mc@q}\\};
if \mc@temp < \EPSILON then {
  % we cannot break for-loop in tikzmath
  \mc@stop = 1;
};
if \mc@stop == 0 then {
  % rotation angle: -pi/4 <=   <= pi/4
  % assuming tan( ) = t, sin( ) = s, cos( ) = c
  % tau = (a[pp]-a[qq])/(2a[pq]) = cot(2 ) =
  ↪  (1-tan( )^2)/2tan( )
  % then t satisfies the quadratic equation:
  % t^2 + 2 tau t - 1 = 0
  % Choosing the root that is smaller in
  ↪  absolute value,
  % we can compute all relevant quantities
  ↪  without
  % using trigonometric functions,
  % t = sign(tau)/(abs(tau)+sqrt(1+tau^2))
```

```
% i.e. t = -tau + sqrt(1+tau^2), if tau >=0;
↪
%      t = -tau - sqrt(1+tau^2), otherwise.
% c = 1/sqrt(1+t^2), s = c*t
real \mc@c, \mc@s, \mc@t, \mc@tau;
\mc@tau = \fpeval{(#2{\mc@p,\mc@p}-#2{\
↪  mc@q,\mc@q})/(2*#2{\mc@p,\mc@q})};
if \mc@tau >= 0 then {
  \mc@t = \fpeval{- \mc@tau +
  ↪  sqrt(1+(\mc@tau)^2)};
} else {
  \mc@t = \fpeval{- \mc@tau -
  ↪  sqrt(1+(\mc@tau)^2)};
};
\mc@c = \fpeval{1 / sqrt(1+(\mc@t)^2)};
\mc@s = \fpeval{\mc@t * \mc@c};
% print{$\tau:\mc@tau,\\
↪  \tan\theta:\mc@t,\\ \cos\theta:\mc@c,\\
↪  sin\theta:\mc@s$\\};
% rotation matrix:
% P = [
%   {...  .  ...  .  ...},
%   {...  c  ... -s  ...},
%   {...  s  ...  c  ...},
%   {...  .  ...  .  ...}
% ],
% perform Jacobi rotation
for \mc@i in {1,...,\mc@n}{
  \mc@ap{\mc@i} = \fpeval{  #2{\mc@i,\mc@p}
  ↪  * \mc@c + #2{\mc@i,\mc@q} * \mc@s};
```

```
    \mc@aq{\mc@i} = \fpeval{- #2{\mc@i,\mc@p}
    ↪  * \mc@s + #2{\mc@i,\mc@q} * \mc@c};
    \mc@vp{\mc@i} = \fpeval{  #3{\mc@i,\mc@p}
    ↪  * \mc@c + #3{\mc@i,\mc@q} * \mc@s};
    \mc@vq{\mc@i} = \fpeval{- #3{\mc@i,\mc@p}
    ↪  * \mc@s + #3{\mc@i,\mc@q} * \mc@c};
  };
  \mc@ap{\mc@p} = #2{\mc@p,\mc@p}*(\mc@c)^2 +
  ↪  #2{\mc@q,\mc@q}*(\mc@s)^2 +
  ↪  2*#2{\mc@p,\mc@q}*\mc@s*\mc@c;
  \mc@aq{\mc@q} = #2{\mc@p,\mc@p}*(\mc@s)^2 +
  ↪  #2{\mc@q,\mc@q}*(\mc@c)^2 -
  ↪  2*#2{\mc@p,\mc@q}*\mc@s*\mc@c;
  \mc@ap{\mc@q} = 0.0;
  \mc@aq{\mc@p} = 0.0;
  for \mc@i in {1,...,\mc@n}{
    % Update matrix
    #2{\mc@i,\mc@p} = \mc@ap{\mc@i};
    #2{\mc@i,\mc@q} = \mc@aq{\mc@i};
    #2{\mc@p,\mc@i} = \mc@ap{\mc@i};
    #2{\mc@q,\mc@i} = \mc@aq{\mc@i};
    % Update eigenvectors
    #3{\mc@i,\mc@p} = \mc@vp{\mc@i};
    #3{\mc@i,\mc@q} = \mc@vq{\mc@i};
  };
  #2{\mc@q,\mc@p} = 0.0;
  % for \mc@i in {1,...,\mc@n} {
  %   for \mc@j in {1,...,\mc@n} {
  %     print{[\mc@i,\mc@j]:\ #2{\mc@i,\mc@j},
  ↪  \ };
```

```
            %   };
            %   print{\\};
            % };
          };%if
        };%if
      };%for
      if \mc@stop == 0 then {
        \status = 0;
      };
    }%tikzmath
  },%jacobi
  % create the matrix of a quadratic form
  % conic/matrix={\q,a,b,c,d,e,f}
  conic/matrix/.code args = {#1,#2,#3,#4,#5,#6,#7} {
    \tikzmath {
      #1{1,1} = #2;     #1{1,2} = #3 / 2; #1{1,3} = #5 /
      ↪ 2;
      #1{2,1} = #3 / 2; #1{2,2} = #4;     #1{2,3} = #6 /
      ↪ 2;
      #1{3,1} = #5 / 2; #1{3,2} = #6 / 2; #1{3,3} = #7;
    }
  },
  % eliminate the cross term in a quadratic form by
  ↪ rotation
  % conic/rotate={\q,\qr,\angle}
  % \qr: the quadratic matrix after rotation
  % \angle: rotation angle
  conic/rotate/.code args = {#1,#2,#3} {
    \tikzmath {
      if abs(#1{1,2}) > \EPSILON then {
```

```
    real \mc@c, \mc@s, \mc@t, \mc@tau;
    \mc@tau =
     ↪  \fpeval{(#1{1,1}-#1{2,2})/(2*#1{1,2})};
    if \mc@tau >= 0 then {
      \mc@t = \fpeval{- \mc@tau +
       ↪  sqrt(1+(\mc@tau)^2)};
    } else {
      \mc@t = \fpeval{- \mc@tau -
       ↪  sqrt(1+(\mc@tau)^2)};
    };
    \mc@c = \fpeval{1 / sqrt(1+(\mc@t)^2)};
    \mc@s = \fpeval{\mc@t * \mc@c};
    #3 = atan(\mc@t);
    % print{\mc@c, \mc@s, angle:#3\\};
    % perform rotation
    #2{1,1} = #1{1,1}*(\mc@c)^2 +
     ↪  #1{1,2}*2*\mc@s*\mc@c + #1{2,2}*(\mc@s)^2;
    #2{1,2} = 0.0;
    #2{1,3} = #1{1,3}*\mc@c + #1{2,3}*\mc@s;
    #2{2,2} = #1{1,1}*(\mc@s)^2 -
     ↪  #1{1,2}*2*\mc@s*\mc@c + #1{2,2}*(\mc@c)^2;
    #2{2,3} = -#1{1,3}*\mc@s + #1{2,3}*\mc@c;
    #2{3,3} = #1{3,3};
    % symetric
    #2{2,1} = #2{1,2};
    #2{3,1} = #2{1,3};
    #2{3,2} = #2{2,3};
  } else {
    % unchanged
    #3 = 0;
```

```
        int \mc@i, \mc@j;
        for \mc@i in {1,2,3} {
          for \mc@j in {1,2,3} {
            #2{\mc@i,\mc@j} = #1{\mc@i,\mc@j};
          };
        };
      };
    }
  },
  % translate the origin of axes to the center(of
  ↪  ellipse or hyperbola)
  % or vertex (of parabola)
  % conic/translate={\q,\qt,\xshift,\yshift}
  % \qt : the quadratic matrix after translation
  % \xshift,\yshift: the center or vertex
  conic/translate/.code args = {#1,#2,#3,#4} {
    \tikzset{
      cofactors={#1,\mc@cofactors,3},
    }
    \tikzmath{
      if abs(\mc@cofactors{3,3}) > \EPSILON then {
        % ellipse or hyperbola
        #3 = \fpeval{\mc@cofactors{3,1} /
        ↪  \mc@cofactors{3,3}};
        #4 = \fpeval{\mc@cofactors{3,2} /
        ↪  \mc@cofactors{3,3}};
      } else {
        % parabola
        if abs(#1{1,1}) > \EPSILON then {
          #3 = \fpeval{- #1{1,3} / #1{1,1}};
```

```
            #4 = \fpeval{((#1{1,3})^2-#1{1,1}*#1{3,3})/
            ↪  (2*#1{1,1}*#1{2,3})};
          } else {
            #3 = \fpeval{((#1{2,3})^2-#1{2,2}*#1{3,3})/
            ↪  (2*#1{2,2}*#1{1,3})};
            #4 = \fpeval{- #1{2,3} / #1{2,2}};
          };%if
        };%if
        % perform translation
        #2{1,1} = #1{1,1}; #2{1,2} = #1{1,2};
        #2{2,1} = #1{2,1}; #2{2,2} = #1{2,2};
        #2{1,3} = #1{1,1}*#3 + #1{1,2}*#4 + #1{1,3};
        #2{2,3} = #1{1,2}*#3 + #1{2,2}*#4 + #1{2,3};
        #2{3,3} = (#3)^2*#1{1,1} + 2*#3*#4*#1{1,2} +
        ↪  2*#3*#1{1,3} + (#4)^2*#1{2,2} + 2*#4*#1{2,3} +
        ↪  #1{3,3};
        % symetric
        #2{3,1} = #2{1,3};
        #2{3,2} = #2{2,3};
      }%tikzmath
    },
    % reduce conic matrix from general to standard form
    % conic/reduce={\q,\qs,\angle,\xshift,\yshift,\q}
    % \q: the quadratic matrix in a general form
    % \qs: the quadratic matrix in a standard form
    % \angle: rotation angle
    % \xshift,\yshift: new origin
    conic/reduce/.code args = {#1,#2,#3,#4,#5} {
      \tikzset {
        conic/rotate={#1,\conic@qr,#3},
```

```
    conic/translate={\conic@qr,\conic@qt,\
    ↪ conic@xshift,\conic@yshift},
}
\tikzmath {
  if abs(\conic@qt{1,1}) > \EPSILON &&
  ↪ abs(\conic@qt{2,2}) > \EPSILON then {
    % ellipse or hyperbola
    \conic@qt{1,1} = \fpeval{-\conic@qt{1,1} /
    ↪ \conic@qt{3,3}};
    \conic@qt{2,2} = \fpeval{-\conic@qt{2,2} /
    ↪ \conic@qt{3,3}};
    \conic@qt{3,3} = -1.0;
  } else {
    % parabola
    if abs(\conic@qt{1,1}) < \EPSILON then {
      \conic@qt{1,3} = \fpeval{\conic@qt{1,3} /
      ↪ \conic@qt{2,2}};
      \conic@qt{3,1} = \conic@qt{1,3};
      \conic@qt{2,2} = 1.0;
    } else {
      \conic@qt{2,3} = \fpeval{\conic@qt{2,3} /
      ↪ \conic@qt{1,1}};
      \conic@qt{3,2} = \conic@qt{2,3};
      \conic@qt{1,1} = 1.0;
    };%if
  };%if
  int \mc@i, \mc@j;
  for \mc@i in {1,2,3} {
    for \mc@j in {1,2,3} {
      #2{\mc@i,\mc@j} = \conic@qt{\mc@i,\mc@j};
```

```
        };
      };
      % restore xshift and yshift in the original
      ↪ coordinate system
      #4 = \conic@xshift * cos(#3) - \conic@yshift *
      ↪ sin(#3);
      #5 = \conic@xshift * sin(#3) + \conic@yshift *
      ↪ cos(#3);
    }%tikzmath
  },
}

\makeatother
```

参考文献

[1] Syntax for path specifications. https://tikz.dev/tikz-paths.

[2] Homothetic center. https://en.wikipedia.org/wiki/Homothetic_center.

[3] Coordinate calculations. https://tikz.dev/tikz-coordinates.

[4] Line-line intersection. https://mathworld.wolfram.com/Line-LineIntersection.html.