matplotlib 笔记本

吴嘉军

2026 年 1 月 23 日

摘要

这里是摘要

关键词：这里是关键词 1、关键词 2

目录

§1、 简单入门

一般来说,matplotlib 需要和 numpy, 搭配着使用, 因为 matplotlib 主要是用来绘图的, 而 numpy 主要是用来进行数值计算的, 两者结合起来可以更好地进行数据处理和可视化.

1、 安装

在使用 matplotlib 之前, 需要先安装它. 可以使用 pip 命令进行安装

```
pip install matplotlib
```

安装完成后, 可以使用以下命令来验证是否安装成功.

1.1、 导入库

在使用 matplotlib 之前, 需要先导入它. 通常情况下, 我们会导入 matplotlib.pyplot 模块, 并将其重命名为 plt. 这样可以方便我们后续的使用.

```python
import matplotlib.pyplot as plt
import numpy as np
```

1.2、 创建数据

在绘图之前, 我们需要先创建一些数据. 通常情况下, 我们会使用 numpy 来创建数据. 例如, 我们可以使用 numpy 的 linspace 函数来创建一个等间隔的数组.

```python
x = np.linspace(0, 10, 100) # 创建一个从0到10的等间隔数组,包含100个点
y = np.sin(x) # 计算x对应的正弦值
```

2、 绘制图形

使用 matplotlib 绘制图形非常简单. 我们只需要调用 plt.plot 函数, 并传入 x 和 y 数据即可. 然后, 使用 plt.show 函数来显示图形.

```python
plt.plot(x, y) # 绘制图形
plt.xlabel('X轴标签') # 设置X轴标签
plt.ylabel('Y轴标签') # 设置Y轴标签
plt.title('图形标题') # 设置图形标题
plt.show() # 显示图形
```

下面是一个简单的例子, 展示如何使用 matplotlib 绘制一个正弦波.

```python
import matplotlib.pyplot as plt
import numpy as np
# 创建数据
x = np.linspace(0, 4 * np.pi, 100)
y = np.sin(x)

# 绘制图形
plt.plot(x, y)
plt.xlabel('X')
plt.ylabel('Y')
plt.title('sin')

plt.savefig('figures/section1/1.1.png') # 保存图形为文件
# Show the plot
plt.show()
```

运行上述代码, 将会显示一个包含正弦波的图形窗口.

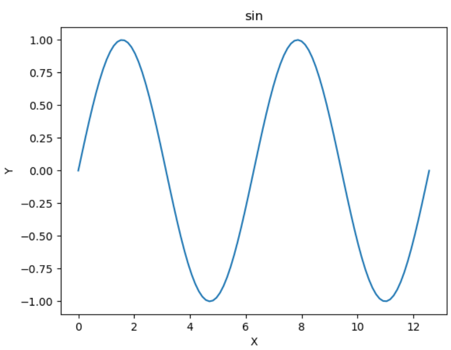

图 1.1: 正弦波图形

3、 插入中文

在使用 matplotlib 绘图时, 如果需要插入中文, 可以使用以下方法:

```python
plt.xlabel('X轴标签', fontproperties='SimHei') # 设置X轴标签为中文
plt.ylabel('Y轴标签', fontproperties='SimHei') # 设置Y轴标签为中文
plt.title('图形标题', fontproperties='SimHei') # 设置图形标题为中文
```

这里的'SimHei' 是黑体字体的名称, 你也可以根据需要选择其他字体. 确保你的系统中已经安装了所需的中文字体.

```python
import matplotlib.pyplot as plt
import numpy as np
# 创建数据
x = np.linspace(0, 4 * np.pi, 100)
y = np.sin(x)

# 绘制图形
plt.plot(x, y)
plt.xlabel('X轴标签', fontproperties='SimHei') # 设置X轴标签为中文
plt.ylabel('Y轴标签', fontproperties='SimHei') # 设置Y轴标签为中文
plt.title('图形标题', fontproperties='SimHei') # 设置图形标题为中文

plt.savefig('figures/section1/1.2.png') # 保存图形为文件
# Show the plot
```

```
plt.show()
```

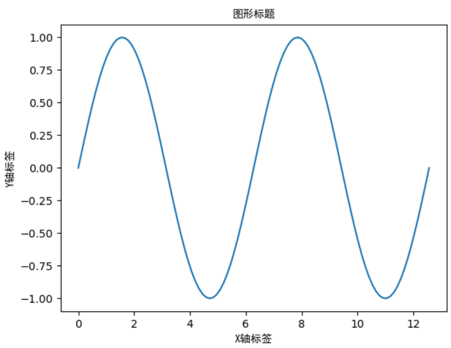

图 1.2

如果觉得太麻烦了, 可以在代码开头添加如下代码, 这样就可以全局使用中文字体了.

```
import matplotlib.pyplot as plt
from matplotlib import rcParams
rcParams['font.sans−serif'] = ['SimHei'] # 全局设置中文字体为黑体
rcParams['axes.unicode_minus'] = False # 解决负号显示问题
```

运行上述代码, 将可以在 fig 中添加中文, 以下代码为例:

```
import matplotlib.pyplot as plt
import numpy as np
# 创建数据
plt.rcParams['font.sans−serif'] = ['SimHei'] # 设置中文字体为SimHei
plt.rcParams['axes.unicode_minus'] = False # 解决负号显示问题
x = np.linspace(0, 4 * np.pi, 100)
y = np.sin(x)

# 绘制图形
plt.plot(x, y)
plt.xlabel('X轴标签') # 设置X轴标签为中文
plt.ylabel('Y轴标签') # 设置Y轴标签为中文
plt.title('图形标题') # 设置图形标题为中文

plt.savefig('figures/section1/1.3.png') # 保存图形为文件
# Show the plot
plt.show()
```

4、 坐标布局

在 matplotlib 中, 可以通过多种方式来调整图形的坐标布局. 以下是一些常用的方法:

```
import matplotlib.pyplot as plt
import numpy as np

plt.rcParams['font.sans−serif'] = ['SimHei'] # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False # 用来正常显示负号
fig = plt.figure("title") # an empty figure with no Axes
fig, ax = plt.subplots() # a figure with a single Axes
fig.suptitle("Figure title") # set the Figure title
ax.set_title("Axes title") # set the Axes title
fig, axs = plt.subplots(2, 2) # a figure with a 2x2 grid of Axes
axs[0, 0].set_title("Axes [0,0] title") # set the title for a specific Axes
axs[1, 1].set_title("Axes [1,1] title")
fig.tight_layout() # adjust spacing between subplots

# a figure with one Axes on the left, and two on the right:
fig, axs = plt.subplot_mosaic([['left', 'right_top'],
                               ['left', 'right_bottom']])
axs['left'].set_title("左侧子图标题")
axs['right_top'].set_title("右上子图标题")
axs['right_bottom'].set_title("右下子图标题")
fig.tight_layout() # adjust spacing between subplots
plt.savefig('figures/section1/1.4.png') # 保存图形为文件
plt.show()
```

运行上述代码, 将会显示一个包含三个子图的图形窗口, 一个在左, 两个在右. 如下所示:

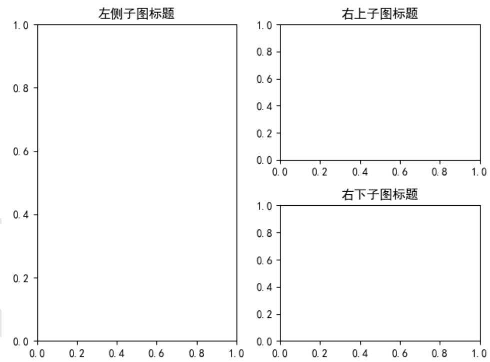

图 1.3: 多个子图布局

5、 坐标轴

在 matplotlib 中, 可以通过多种方式来调整图形的坐标轴. 以下是一些常用的方法:

```
import matplotlib.pyplot as plt
import numpy as np

plt.rcParams['font.sans−serif'] = ['SimHei'] # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False # 用来正常显示负号

fig,ax = plt.subplots()
x = np.arange(0, 4*np.pi, 0.01)
y = np.sin(x)
ax.plot(x, y)
# 设置坐标轴标签
ax.set_xlabel("X轴标签")
ax.set_ylabel("Y轴标签")
# 设置坐标轴范围
ax.set_xlim(0, 9)
ax.set_ylim(−1, 1)
# 设置刻度
ax.set_xticks([0, 0.5*np.pi, np.pi, 1.5*np.pi, 2*np.pi, 2.5*np.pi,3*np.pi,3.5*np.pi,4*np.pi])
ax.set_yticks([−1, −0.8, −0.6, −0.4, −0.2, 0, 0.2, 0.4, 0.6, 0.8, 1])

# 设置刻度标签
# ax.set_xticklabels(['0', '零点五', '', '一点五', '2', '二点五','3','三点五','4'])
# ax.set_yticklabels(['−1', '−0.8', '−0.6', '−0.4', '−0.2', '0', '0.2', '0.4', '0.6', '0.8', '1'])
# 设置标题
ax.set_title("坐标轴示例")
```

```
# 显示网格
ax.grid(True)
# 上下左右边框可见性
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
# 设置刻度方向、长度、宽度和颜色
ax.tick_params(direction='inout', length=10, width=10, colors='red')
# 设置刻度线位置
ax.xaxis.set_ticks_position('bottom')
ax.yaxis.set_ticks_position('left')
# 设置刻度线样式
# 打开次刻度线
# 关闭次刻度线
ax.minorticks_off()
# 显示主次网格线
ax.grid(which='minor', linestyle='--', linewidth=0.5)
ax.minorticks_on()
ax.xaxis.set_tick_params(which='major', size=10, width=1, direction='out',colors='blue')
ax.xaxis.set_tick_params(which='minor', size=5, width=1, direction='out', colors='red')
ax.yaxis.set_tick_params(which='major', size=10, width=0.5, direction='out')
ax.yaxis.set_tick_params(which='minor', size=5, width=1, direction='in', colors='green')

# 设置坐标轴0位置
ax.spines['bottom'].set_position(('data', 0)) # x轴位置
ax.spines['left'].set_position(('data', 0)) # y轴位置

# 添加箭头 ,clip_on=False的作用是防止箭头被裁剪掉
ax.plot(1, 0, '>r',transform=ax.get_yaxis_transform(),clip_on=False) # transform=ax.get_yaxis_transform()表示箭头在y轴上
ax.plot(0, 1, '^b', transform=ax.get_xaxis_transform(),clip_on=False) # transform=ax.get_xaxis_transform()表示箭头在x轴上

plt.savefig('figures/section1/1.5.png') # 保存图形为文件
plt.show()
```

运行上述代码, 将会显示一个包含自定义坐标轴的图形窗口. 如下所示:

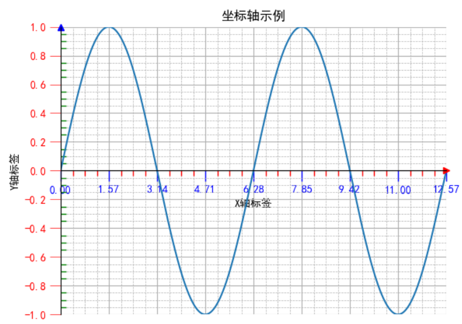

图 1.4: 自定义坐标轴

6、 箭头

在 matplotlib 中, 可以通过多种方式来添加箭头. 以下是一些常用的方法:

```
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches

fig, ax = plt.subplots()

# 弯箭头示例
arrow = mpatches.FancyArrowPatch(
    # 起点和终点坐标
    (0.2, 0.2), (.8, .8),
    # 箭头样式和属性
    arrowstyle='->,head_length=0.4,head_width=0.2',
    connectionstyle='arc3,rad=0.3', # 弯曲半径
    # 其他属性,箭头颜色,线宽,大小等
    color='blue', linewidth=2, mutation_scale=20,
    # 坐标系是data坐标系,不裁剪
    transform=ax.transData, clip_on=False
)
ax.add_patch(arrow)

plt.savefig('figures/section1/1.6.png') # 保存图形为文件
plt.show()
```

运行上述代码, 将会显示一个包含箭头的图形窗口. 如下所示:

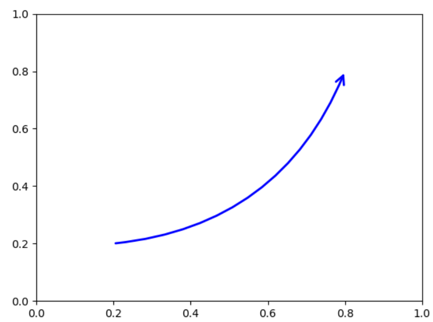

图 1.5: 箭头示例

再举例说明一下如何添加不同样式的箭头.

```
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import numpy as np

plt.rcParams['font.sans-serif'] = ['SimHei'] # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False # 用来正常显示负号

fig, ax = plt.subplots()
x = np.linspace(0, 4*np.pi, 100)
y = np.sin(x)
ax.set_title("箭头示例2")
ax.plot(x, y)
# 直箭头示例
arrow1 = mpatches.FancyArrowPatch(
    (0.7*np.pi, 1.2), (0.5 * np.pi, 1),
    arrowstyle='->', mutation_scale=20,
    connectionstyle='arc3,rad=0.3',
    color='red',
    transform=ax.transData, clip_on=False
)
arrow2 = mpatches.FancyArrowPatch(
    (3.7*np.pi, -1.2), (3.5 * np.pi, -1),
    arrowstyle='->', mutation_scale=20,
    connectionstyle='arc3,rad=0.3',
    color='red',
    transform=ax.transData, clip_on=False
```

```
)

ax.add_patch(arrow1)
ax.add_patch(arrow2)

ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['bottom'].set_position(('data', 0)) # x轴位置
ax.spines['left'].set_position(('data', 0)) # y轴位置

# 添加箭头 ,clip_on=False的作用是防止箭头被裁剪掉
ax.plot(1, 0, '>r',transform=ax.get_yaxis_transform(),clip_on=False) # transform=ax.get_yaxis_transform()表示箭头在y轴上
ax.plot(0, 1, '^b', transform=ax.get_xaxis_transform(),clip_on=False) # transform=ax.get_xaxis_transform()表示箭头在x轴上

# 点一个( /2,1)位置
ax.plot(0.5*np.pi, 1, 'og', markersize=8)
ax.plot(3.5*np.pi, -1, 'ob', markersize=8)

# 标注文字说明
ax.text(0.7*np.pi - 0.1, 1.2 - 0.1, "最高点($\pi$/2,1)", color='red', fontsize=10)
ax.text(3.7*np.pi - 0.1, -1.2 - 0.1, "最低点(7$\pi$/2,1)", color='red', fontsize=10)

plt.savefig('figures/section1/1.7.png') # 保存图形为文件
plt.show()
```

运行上述代码, 将会显示一个包含不同样式箭头的图形窗口. 如下所示:

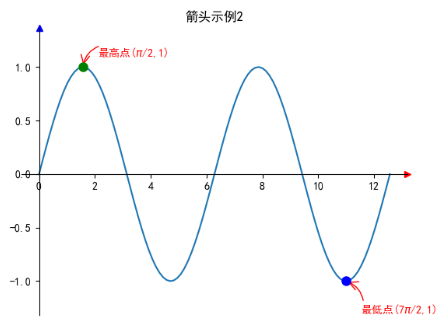

图 1.6: 不同样式箭头示例

7、 文字说明

在 matplotlib 中, 可以通过多种方式来添加文字说明. 以下是一些常用的方法:

```
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import numpy as np

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False


fig,ax = plt.subplots()
fig.suptitle("文字说明")
ax.set_title("latex文字说明")

ax.plot([0.5,1,1.5,2,2.5],[-0.5,-0.25,0,0.25,0.5],'og', markersize=2)
# 添加文字说明
ax.text(0.5, -0.5, "matplotlib可以插入latex公式", color='green', fontsize=12)
ax.text(1, -0.25, r"$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}$", color='green', fontsize=15)
ax.text(1.5, 0, r"$\int_a^b f(x) \mathrm{d}x = F(b) - F(a)$", color='green', fontsize=15)
ax.text(2, 0.25, r"$e^{i \pi} + 1 = 0$", color='green', fontsize=15)
ax.text(2.5, 0.5, r"$\nabla \cdot \mathbf{E} = \frac {\rho} {\varepsilon_0}$", color='green', fontsize=15)



ax.set_xlim(0,3.5)
ax.set_ylim(-1, 1)

ax.grid(True)
plt.savefig('figures/section1/1.8.png') # 保存图形为文件
plt.show()
```

运行上述代码, 将会显示一个包含文字说明的图形窗口. 如下所示:

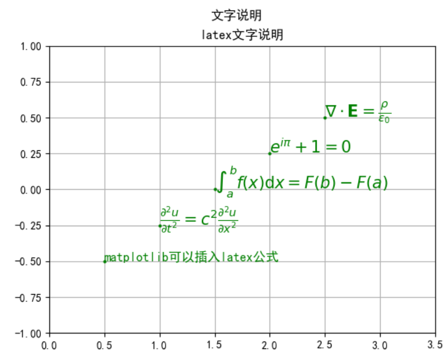

图 1.7: 文字说明示例

§2、 patches

在 matplotlib 中,patches 模块提供了许多用于绘制图形元素的类. 这些类可以用来创建各种形状, 如矩形、圆形、椭圆等. 以下是一些常用的 patches 类:

- `Rectangle`: 用于绘制矩形.
- `Circle`: 用于绘制圆形.
- `Ellipse`: 用于绘制椭圆.
- `Polygon`: 用于绘制多边形.

例如, 我们可以使用 patches.Rectangle 来创建一个正方形:

```python
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import numpy as np

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

fig, ax = plt.subplots()
# 创建一个正方形
rec = mpatches.Rectangle((0.2,0.2), 0.6, 0.6, facecolor='blue', edgecolor='black')
ax.add_patch(rec)

ax.plot(0.2,0.2,color = 'red', marker = 'o', markersize=10)

ax.set_xlim(0,1)
ax.set_ylim(0,1)
ax.set_title('正方形示例')
# x轴和y轴一样长
ax.set_aspect('equal')
# 保持图形
plt.savefig('figures/section2/2.1.png')
plt.show()
```

运行上述代码, 将会显示一个包含正方形的图形窗口. 如下所示:

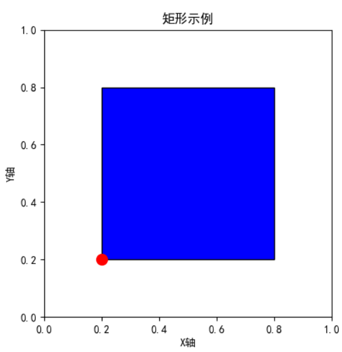

图 2.1: 正方形示例

1、 矩形

上面的例子已经展示了如何使用 *patches.Rectangle* 来创建一个矩形. 我们还可以通过设置不同的参数来调整矩形的属性, 如颜色、边框宽度等.

```python
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import numpy as np

plt.rcParams["font.sans-serif"] = ['SimHei']
plt.rcParams["axes.unicode_minus"] = False
# 创建子图
fig,ax = plt.subplots(3,2,figsize=(10,15))

# angle=0表示不旋转,facecolor表示填充颜色,edgecolor表示边框颜色,linewidth表示边框宽度,alpha表示透明度
rect = mpatches.Rectangle((0.2,0.2),0.6,0.6,angle=0,facecolor="lightblue",edgecolor="blue",linewidth=2,alpha=0.7)
ax[0,0].add_patch(rect)
ax[0,0].set_xlim(0,1)
ax[0,0].set_ylim(0,1)
# adjustable参数设置为box时，长宽比固定
ax[0,0].set_aspect('equal', adjustable='box')

# angle=45表示旋转45度,默认是绕着左下角旋转
rect = mpatches.Rectangle((0.2,0.2),0.6,0.6,angle=45,facecolor="lightgreen",edgecolor="green",linewidth=2,alpha=0.7)
ax[0,1].add_patch(rect)
ax[0,1].set_xlim(0,1)
ax[0,1].set_ylim(0,1)
ax[0,1].set_aspect('equal', adjustable='box')

# 填充颜色和边框颜色,填充无颜色,边框填充红色
rect = mpatches.Rectangle((0.2,0.2),0.6,0.6,angle=0,facecolor='none',edgecolor="red",linewidth= 2,alpha=1)
ax[1,0].add_patch(rect)
ax[1,0].set_xlim(0,1)
ax[1,0].set_ylim(0,1)
ax[1,0].set_aspect('equal', adjustable='box')
# 修改长方形的位置在原点,默认是以左下角为原点
rect = mpatches.Rectangle((0,0),0.6,0.6,angle=0,facecolor="orange",edgecolor="brown",linewidth=2,alpha=0.7)
ax[1,1].add_patch(rect)
ax[1,1].set_xlim(0,1)
ax[1,1].set_ylim(0,1)
ax[1,1].set_aspect('equal', adjustable='box')
ax[1,1].plot(0,0,'ro',markersize = 10) # 标记原点位置

# 修改长方形的长和宽,宽为1,长为0.6
rect = mpatches.Rectangle((0.0,0.1),1,0.6,angle=0,facecolor="purple",edgecolor="none",linewidth=2,alpha=0.7)
ax[2,0].add_patch(rect)
ax[2,0].set_xlim(0,1)
ax[2,0].set_ylim(0,1)
ax[2,0].set_aspect('equal', adjustable='box')

# 修改长方形的透明度,alpha=0.3表示透明度为30%
rect1 = mpatches.Rectangle((0.2,0.2),0.6,0.6,angle=0,facecolor="red",edgecolor="none",linewidth=2,alpha=0.3)
rect2 = mpatches.Rectangle((0.3,0.3),0.6,0.6,angle=0,facecolor="blue",edgecolor="none",linewidth=2,alpha=0.4)
ax[2,1].add_patch(rect1)
ax[2,1].add_patch(rect2)
ax[2,1].set_xlim(0,1)
ax[2,1].set_ylim(0,1)
ax[2,1].set_aspect('equal', adjustable='box')

```

```
plt.savefig("figures/section2/2.2.png",dpi=300)
plt.show()
```

运行上述代码, 将会显示一个包含自定义矩形的图形窗口. 如下所示:

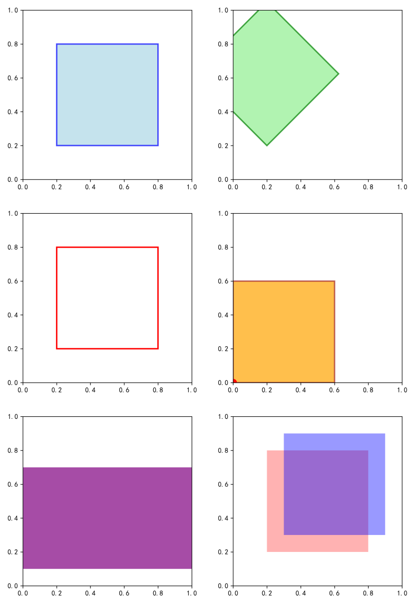

图 2.2: 自定义矩形示例

2、 三角形

我们可以使用 *patches.Polygon* 来创建一个三角形. 以下是一个示例:

```
import matplotlib.pyplot as plt
import matplotlib.patches as mpathes
import numpy as np

plt.rcParams['font.sans−serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

# 创建子图
fig, ax = plt.subplots(1, 2, figsize=(10,5))
# 创建一个等边三角形,中心在(0.5,0.5),边长为0.8,RegularPolygon表示正多边形, numVertices=3表示三边形,radius表示外接圆半径,orientation表示旋转角度 pi/2表示旋转90度
triangle = mpathes.RegularPolygon((0.5, 0.5), numVertices=3, radius=0.4, orientation=np.pi / 2,
                                  facecolor='lightcoral', edgecolor='darkred', linewidth=2, alpha=0.7)
ax[0].add_patch(triangle)
ax[0].set_xlim(0, 1)
ax[0].set_ylim(0, 1)
ax[0].set_aspect('equal')
ax[0].set_title('等边三角形')

# 创建一个直角三角形,顶点坐标为(0.2,0.2),(0.8,0.2),(0.8,0.8)
right_triangle = mpathes.Polygon([(0.2, 0.2), (0.8, 0.2), (0.8, 0.8)], facecolor='lightblue', edgecolor='darkblue', linewidth=2, alpha=0.7)
ax[1].add_patch(right_triangle)
ax[1].set_xlim(0, 1)
ax[1].set_ylim(0, 1)
ax[1].set_aspect('equal')
ax[1].set_title('直角三角形')

plt.savefig("figures/section2/2.3.png", dpi=300)
plt.show()
```

运行上述代码, 将会显示一个包含三角形的图形窗口. 如下所示:

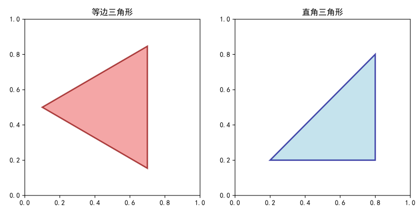

图 2.3: 三角形示例

这样绘制的三角形是已知顶点坐标的. 如果是只知道两条边长和其中的夹角, 可以通过计算得到其他两个顶点的坐标.

2.1、 边角边画三角形

举个例子, 假设已知三角形的一条边长为 a, 另一条边长为 b, 夹角为 θ, 则可以通过以下代码计算出其他两个顶点的坐标并绘制三角形:

```
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import numpy as np

plt.rcParams['font.sans−serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

# 创建子图
fig, ax = plt.subplots(1, 1, figsize=(10,5))
# 知道三角形的两条边长和夹角,计算顶点坐标
a = 0.6 # 边长a
b = 0.5 # 边长b
theta = np.radians(60) # 夹角60度转换为弧

# 计算顶点坐标
A = [0.2, 0.2]
B = [A[0] + a, A[1]]
C = [A[0] + b * np.cos(theta), A[1] + b * np.sin(theta)]
# 创建三角形
triangle = mpatches.Polygon([A, B, C], facecolor='lightcoral', edgecolor='darkred', linewidth=2, alpha=0.7)
ax.add_patch(triangle)
ax.set_xlim(0, 1)
ax.set_ylim(0, 1)
ax.set_aspect('equal')
```

```python
ax.set_title('已知两边和夹角的三角形')
# *A表示解包坐标,*解包操作符,表示将元组拆开,写字在旁边,
ax.plot(*A, 'ro', label='顶点A') # 标记顶点A
ax.plot(*B, 'go', label='顶点B') # 标记顶点B
ax.plot(*C, 'bo', label='顶点C') # 标记顶点C

ax.text(*A, 'A', fontsize=12, ha='right', va='bottom')
ax.text(*B, 'B', fontsize=12, ha='left', va='bottom')
ax.text(*C, 'C', fontsize=12, ha='center', va='bottom')
ax.legend()

plt.savefig("figures/section2/2.4.png", dpi=300)
plt.show()
```

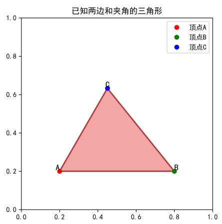

图 2.4: 三角形示例 2

2.2、 向量

还可以使用向量在图像中需找特殊点, 测量边长, 测量角, 绘制角平分线.

```python
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import numpy as np

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

a = 1
b = 1
theta = np.radians(60) # 夹角60度转换为弧
A = [0.2, 0.2]
B = [A[0] + a, A[1]]
C = [A[0] + b * np.cos(theta), A[1] + b * np.sin(theta)]
A = np.array(A)
B = np.array(B)
C = np.array(C)
print(f'{A},{B},{C}')
fig,ax = plt.subplots()
ax.text(A[0],A[1],"A")
ax.text(B[0],B[1],"B")
ax.text(C[0],C[1],"C")
triangle = mpatches.Polygon([A, B, C], facecolor='lightblue', edgecolor='
    darkred', linewidth=2, alpha=0.7)

vec_ab = B - A
vec_ac = C - A
vec_bc = B - C
print(vec_ab)
ab = np.linalg.norm(vec_ab)
print(f'边长ab:{ab}')

mid_ab = (A + B) / 2
print(f'中点坐标:{mid_ab}')

angle_A = np.dot(vec_ab,vec_ac) / (np.linalg.norm(vec_ab) * np.linalg.
    norm(vec_ac))
# 计算弧度角
angle_radians = np.arccos(angle_A)
# 转换为度数
angle_degrees = np.degrees(angle_radians)
print(f"A夹角{angle_degrees}")

# 计算角平分线
# 归一化向量
unit_vec_ab = vec_ab / np.linalg.norm(vec_ab)
unit_vec_ac = vec_ac / np.linalg.norm(vec_ac)
print(unit_vec_ab)
print(unit_vec_ac)


# # 计算角平分线向量
bisector = unit_vec_ab + unit_vec_ac + A
ax.text(bisector[0],bisector[1],'角平分线')
x, y = zip(A,bisector) # 使用zip解包坐标
print(x)
print(y)
plt.plot(x,y,'bo',linestyle = '-',label = '角平分线')

angle_A = np.dot(vec_ab,bisector) / (np.linalg.norm(vec_ab) * np.linalg.
    norm(bisector))
# 计算弧度角
angle_radians = np.arccos(angle_A)
# 转换为度数
angle_degrees = np.degrees(angle_radians)
print(f"A夹角{angle_degrees}")

# # 计算中线
mid_bc = (B + C) / 2
ax.text(mid_bc[0],mid_bc[1],'中点')
print(mid_bc)
x, y = zip(A, mid_bc) # 使用zip解包坐标
# print(x)
# print(y)
plt.plot(x,y,'ro',linestyle = '-',label = '中线')



ax.add_patch(triangle)
ax.set_aspect('equal')
plt.savefig('figures/section2/2.5.png',dpi=300)
plt.show()
```

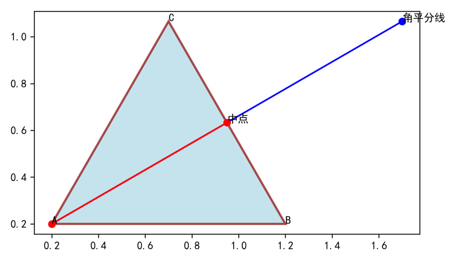

图 2.5: 三角形示例 3

2.3、 三边画三角形

如果知道三条边的长度 a,b,c, 也可以用于绘制三角形.

```python
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import numpy as np

a,b,c = 1,1,1
A = [0.,0.]
# 利用余弦定理
# cosA=(b^2+c^2−a^2)/2bc
thetaA = (b**2 + c**2 - a**2) / (2*b*c)
angle_radians = np.arccos(thetaA)
# 转换为度数
angle_degrees = np.degrees(angle_radians)

print(f"{angle_degrees:.2f}")
B = [A[0] + c, A[1]]
C = [A[0] + b * np.cos(angle_radians), A[1] + b * np.sin(angle_radians)]


fig,ax = plt.subplots()
triangle = mpatches.Polygon([A, B, C], facecolor='lightblue', edgecolor='darkred', linewidth=2, alpha=0.7)
ax.add_patch(triangle)
ax.text(0.5,0.5,r"$ \cos A = \frac{b^2 + c^2 - a^2}{2bc}$",fontsize = 18)
ax.text(*A,'$A$',color = "red")
ax.text(*B,'$B$')
ax.text(*C,'$C$',color = 'blue')

ax.set_xlim(0,c + 0.1)
ax.set_ylim(0,b)
ax.set_aspect('equal')

plt.savefig("figures/section2/2.6.png")
plt.show()
```

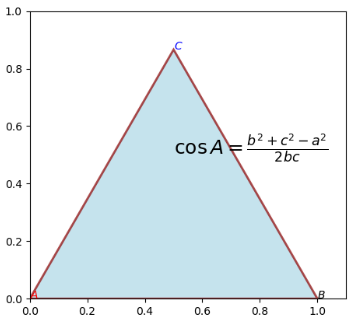

图 2.6: 三角形示例 4

2.4、 角边角画三角形

知道三角形的两个角，和两角中间的边长，也可以绘制三角形. 可以运用正弦定理:

$\frac{a}{\sin A} = \frac{b}{\sin B} = \frac{c}{\sin C}$ 用下面的例子绘图:

```python
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.patches as mpatches

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

# 初始条件
thetaA,thetaB = 30,90
c = 1
A = [0.,0.]

thetaC = 180 - thetaA - thetaB
if thetaC < 0:
    print("构造不了三角形")
    pass

angle_radiansA = np.radians(thetaA)
angle_radiansB = np.radians(thetaB)
angle_radiansC = np.radians(thetaC)

a = np.sin(angle_radiansA) / (np.sin(angle_radiansC) / c)
b = np.sin(angle_radiansB) / (np.sin(angle_radiansC) / c)

print(f"{a},{b}")

B = [A[0] + c, A[1]]
C = [A[0] + b * np.cos(angle_radiansA), A[1] + b * np.sin(angle_radiansA)]

fig,ax = plt.subplots()
triangle = mpatches.Polygon([A, B, C], facecolor='lightblue', edgecolor='darkred', linewidth=2, alpha=0.7)
ax.add_patch(triangle)
ax.set_xlim(0,c + 0.1)
ax.set_ylim(0,b)
ax.text(0.5,0.3,r"$\frac{a}{\sin A} = \frac{b}{\sin B} = \frac{c}{\sin C}$",fontsize = 18)
ax.text(*A,'$A$',color = "red")
ax.text(*B,'$B$')
ax.text(*C,'$C$',color = 'blue')
ax.set_aspect('equal')
plt.savefig("figures/section2/2.7.png")
plt.show()
```

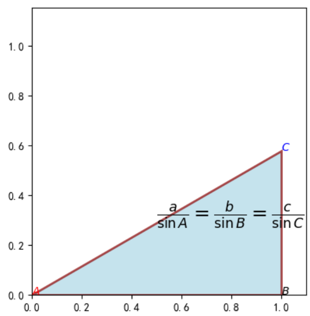

图 2.7: 三角形示例 5

3、 圆弧

```python
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.patches as mpatches

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

A = [0.,0.]
B = [1.,0.]
theta = 120
```

```python
radians = np.radians(theta)

C = [A[0] + 1. * np.cos(radians),A[1] + 1. * np.sin(radians)]

fig,ax = plt.subplots()
x,y = zip(A,B)
plt.plot(x,y,'r-')
x,y = zip(A,C)
plt.plot(x,y,'r-')
# 绘制弧线,width和height是椭圆形的长边和短边,相等时相当于圆形弧线,theta1和theta2分别是起点和终点角度
arc = mpatches.Arc((0,0),width=0.3,height=0.3,angle=0,theta1=0,theta2=theta,linewidth = 2,color = "blue")

ax.add_patch(arc)
ax.text(0.1,0.1,fr'$\alpha = {theta}^o$',fontsize = 12)
ax.set_aspect('equal')
plt.savefig('figures/section2/2.8.png')
plt.show()
```

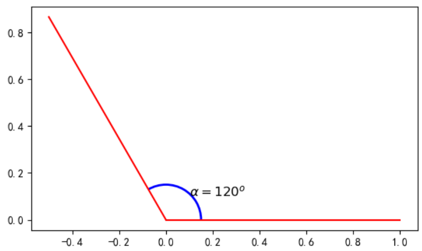

图 2.8: 圆弧

4、 圆形和椭圆

```python
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import numpy as np
import os

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

fig,ax = plt.subplots()

ellipse = mpatches.Ellipse((0,0),1.,0.6,fc = 'red',ec = 'blue')
circle = mpatches.Circle((0,0),0.5,fc = 'yellow',alpha = 0.5,ec = 'red')
regularPolygon = mpatches.RegularPolygon((0,0),numVertices=5,radius=0.5,alpha = 0.5)
ax.add_patch(ellipse)
ax.add_patch(circle)
ax.add_patch(regularPolygon)
ax.set_xlim(-1,1)
ax.set_ylim(-1,1)
ax.set_aspect('equal')
# 文件保存
script_name = os.path.splitext(os.path.basename(__file__))[0]
plt.savefig("figures/section2/" + script_name + '.png')
plt.show()
```

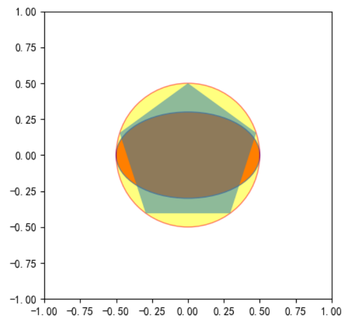

图 2.9: 圆形和椭圆

5、 扇形

```python
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import os

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
fig, ax = plt.subplots()

# 使用 patches 自定义样式
wedge0 = mpatches.Wedge((0.,0.),0.5,theta1=0,theta2=30,linewidth=2, edgecolor='purple', facecolor='plum', alpha=0.7)
wedge1 = mpatches.Wedge((0.,0.),0.5,theta1=30,theta2=60,linewidth=2, edgecolor='green', facecolor='purple', alpha=0.7)
wedge2 = mpatches.Wedge((0.,0.),0.5,theta1=60,theta2=150,linewidth=2, edgecolor='black', facecolor='green', alpha=0.7)
wedge3 = mpatches.Wedge((0.,0.),0.5,theta1=150,theta2=290,linewidth=2, edgecolor='blue', facecolor='yellow', alpha=0.7)
wedge4 = mpatches.Wedge((0.,0.),0.5,theta1=290,theta2=360,linewidth=2, edgecolor='plum', facecolor='red', alpha=0.7)

ax.add_patch(wedge0)
ax.add_patch(wedge1)
ax.add_patch(wedge2)
ax.add_patch(wedge3)
ax.add_patch(wedge4)
ax.set_xlim(-1,1)
ax.set_ylim(-1,1)
ax.set_aspect('equal')
# 文件保存
script_name = os.path.splitext(os.path.basename(__file__))[0]
plt.savefig("figures/section2/" + script_name + '.png')
plt.show()
```

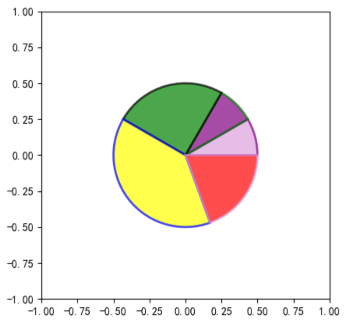

图 2.10: 扇形

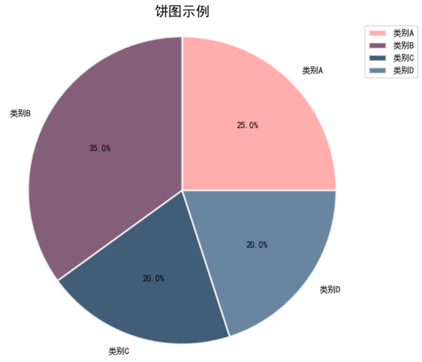

图 2.11: 扇形饼图

plt 也可以绘制扇形饼图

```python
# 利用matplotlib.pyplot也可以直接绘制扇形饼图
import matplotlib.pyplot as plt
# import matplotlib.patches as mpatches
import os

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
# 准备数据
labels = ['类别A', '类别B', '类别C', '类别D']
sizes = [25, 35, 20, 20]
colors = ["#ff9999","#663758","#123456","#456789"]

# 创建图形
fig, ax = plt.subplots(figsize=(8, 6))

# 绘制饼图并获取扇形patch
wedges, texts, autotexts = ax.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%')

# 自定义每个扇形的样式
for i, wedge in enumerate(wedges):
    # 设置边框颜色和宽度
    wedge.set_edgecolor('white')
    wedge.set_linewidth(2)
    # 设置透明度
    wedge.set_alpha(0.8)

# 添加标题
ax.set_title('饼图示例', fontsize=16, fontweight='bold')

# 确保饼图是圆形
ax.axis('equal')

# 添加图例
ax.legend()

# 调整布局
plt.tight_layout()

# 文件保存
script_name = os.path.splitext(os.path.basename(__file__))[0]
plt.savefig("figures/section2/" + script_name + '.png')
plt.show()
```

§3、 隐函数

matplotlib 也可以按照隐函数进行绘图. 这里主要提供两种方法进行画图

1、 圆锥曲线标准方程

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}=1$$

```python
import numpy as np
import matplotlib.pyplot as plt
import os

# 设置中文字体，例如使用“SimHei”
plt.rcParams['font.sans-serif'] = ['SimHei'] # 添加'SimHei'到字体列表
plt.rcParams['axes.unicode_minus'] = False # 解决负号显示为 的问题
fig,ax = plt.subplots()
fig.suptitle(r'隐函数$\frac{x^2}{4} + \frac{y^2}{1} = 1$')
# 定义x和y的范围
x = np.linspace(-3,3, 100)
y = np.linspace(-1.5,1.5, 100)
X, Y = np.meshgrid(x, y)

# 计算Z值，Z = x^2/4 + y^2/1 - 1
Z = X**2/4 + Y**2/1 - 1

# 绘制等高线图
ax.contour(X, Y, Z, levels=[0], colors='black')
ax.plot(1,0, '>r',transform=ax.get_yaxis_transform(),clip_on=False) # transform=ax.get_yaxis_transform()表示箭头在x轴上
ax.plot(0,1, '^b',transform=ax.get_xaxis_transform(),clip_on=False) # transform=ax.get_yaxis_transform()表示箭头在y轴上
ax.set_aspect(1)
ax.set_title(r'隐函数$\frac{x^2}{4} + \frac{y^2}{1} = 1$')


ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
#改变坐标轴的位置
ax.spines['bottom'].set_position(('data', 0))
ax.spines['left'].set_position(('data', 0))
# 文件保存
script_name = os.path.splitext(os.path.basename(__file__))[0]
plt.savefig("figures/section3/" + script_name + '.png')
plt.show()
```

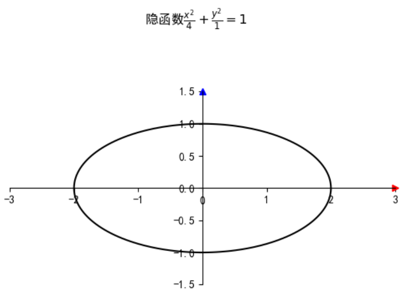

图 3.1: 椭圆

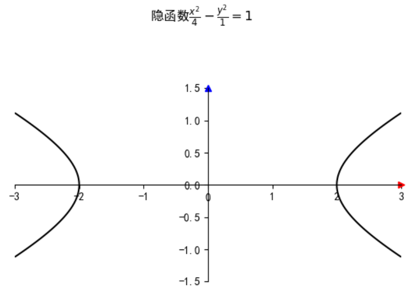

图 3.2: 双曲线

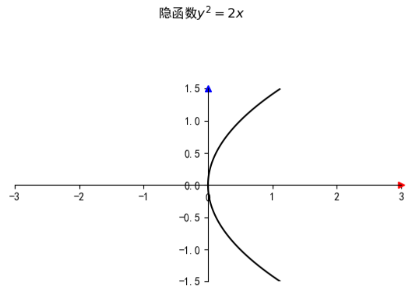

图 3.3: 抛物线

以上可以看出画图的方式是用到了 contour 绘制等高线的方式. 下面我们还可以用到 sympy 库, 利用 sympy 可以直接解出 y 与 x 的关系表达式:$y=f(x)$

2、 圆锥曲线一般方程

圆锥曲线一般方程:

$$Ax^2+Bxy+Cy^2+Dx+Ey+F=0$$

用圆锥曲线的一般方程来绘制一些图片

```python
import sympy as sp
import matplotlib.pyplot as plt
import numpy as np
import os

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

fig,ax = plt.subplots()
ax.set_title('圆锥曲线的一般方程')
A,B,C,D,E,F,x,y = sp.symbols('A,B,C,D,E,F,x,y')
expr = A*x**2 + B * x * y + C * y**2 +D * x + E * y + F
eq = sp.Eq(expr,0)
# 如果圆锥曲线在原点,则没有x和y的一次项,也就是D和E为0
```

```python
subs_dict = {A:-3,B:4,C:-4,D:0,E:0,F:4}
result = eq.subs(subs_dict)

# 解出y关于x的表达式
solutions = sp.solve(result, y)

print(sp.latex(solutions))

# 绘制每个解
x_vals = np.linspace(-10, 10, 2000)
for i, sol in enumerate(solutions):
    # 将符号表达式转换为数值函数
    y_func = sp.lambdify(x, sol, 'numpy')

    try:
        y_vals = y_func(x_vals)
        # 只绘制实数部分
        mask = np.isreal(y_vals)
        if np.any(mask):
            ax.plot(x_vals[mask], np.real(y_vals[mask]),color = 'red',
                    label=f'解 {i+1}', linewidth=2)
    except Exception as e:
        print(f"绘制解 {i+1} 时出错: {e}")
        continue

ax.text(0.,0.,f'${sp.latex(result)}$',
        bbox={'boxstyle': 'round',
              'facecolor': 'lightblue',
              'alpha': 0.7,
              'pad': 0.5}
        ,alpha = 0.8)

offset_x = 0.5 # x方向的偏移量（向左为负）
offset_y = 0.5 # y方向的偏移量（向下为负）
text_x = ax.get_xlim()[0] + offset_x # 获取x轴最小值并加上偏移量
text_y = ax.get_ylim()[0] + offset_y # 获取y轴最小值并加上偏移量

ax.text(text_x,text_y,f'$y={sp.latex(solutions[0])}$',fontsize = 12)
ax.text(text_x,text_y + offset_y,f'$y={sp.latex(solutions[1])}$',fontsize =
        12)

# 文件保存
script_name = os.path.splitext(os.path.basename(__file__))[0]
plt.savefig("figures/section3/" + script_name + '.png')
plt.show()
```

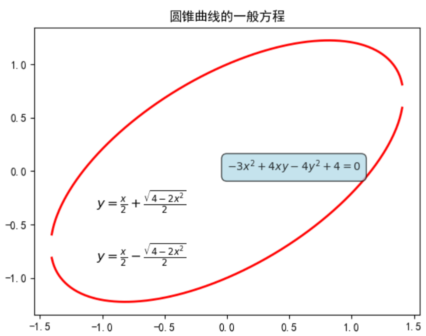

图 3.4: 圆锥曲线一般方程

3、 圆锥曲线和直线

```python
import matplotlib.pyplot as plt
import sympy as sp
import numpy as np
import os

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
fig,ax = plt.subplots()

x,y = sp.symbols("x y")
expr1 = x + y + 1
expr2 = x**2 + y**2 + 3*x + 4*y

eq1,eq2 = sp.Eq(expr1,0),sp.Eq(expr2,0)
results = sp.solve([eq1,eq2],[x,y])

y1 = sp.solve(eq1,y)
y2 = sp.solve(eq2,y)

# 绘制每个解
x_vals = np.linspace(-5, 5, 2000)
for i, sol in enumerate(y1):
    y_func = sp.lambdify(x, sol, 'numpy')
    y_vals = y_func(x_vals)
    ax.plot(x_vals, y_vals,color = 'purple')


for i, sol in enumerate(y2):
    y_func = sp.lambdify(x, sol, 'numpy')
    y_vals = y_func(x_vals)
    ax.plot(x_vals, y_vals,color = 'blue')


# 寻找交点
for result in results:
    plt.plot(*result,'ro')

# 文件保存
script_name = os.path.splitext(os.path.basename(__file__))[0]
plt.savefig("figures/section3/" + script_name + '.png')
plt.show()
```

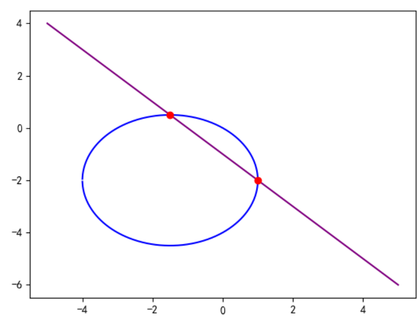

图 3.5: 圆锥曲线和直线

```python
import matplotlib.pyplot as plt
import sympy as sp
import numpy as np
import os

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

fig,ax = plt.subplots()

x,y = sp.symbols("x y",real = True)
```

```python
expr1 = y**2-(x**2 + x + 2)
expr2 = -y + 2

eq1,eq2 = sp.Eq(expr1,0),sp.Eq(expr2,0)
results = sp.solve([eq1,eq2],[x,y])

y1 = sp.solve(eq1,y)
y2 = sp.solve(eq2,y)

# 绘制每个解
x_vals = np.linspace(-3, 3, 2000)
for i, sol in enumerate(y1):
    y_func = sp.lambdify(x, sol, 'numpy')
    y_vals = y_func(x_vals)
    ax.plot(x_vals, y_vals,color = 'purple',label = f"${sp.latex(eq1)}$")

for i, sol in enumerate(y2):
    # 将符号表达式转换为数值函数
    y_func = sp.lambdify(x, sol, 'numpy')
    try:
        y_vals = y_func(x_vals)
        ax.plot(x_vals, y_vals,color = 'red',
                linewidth=2,label = f"${sp.latex(eq2)}$")
    except Exception as e:
        ax.plot(x_vals, [y_vals] * len(x_vals),color = 'red',
                linewidth=2,label = f"${sp.latex(eq2)}$")
        continue

# 绘制交点
# 绘制交点
if results:
    for point in results:
        x_coord, y_coord = point
        ax.plot(x_coord, y_coord, 'ro', markersize=8)
        ax.annotate(f'({x_coord:.2f}, {y_coord:.2f})',
                    (x_coord, y_coord),
                    xytext=(10, 10),
                    textcoords='offset points',
                    bbox=dict(boxstyle='round,pad=0.3', facecolor='yellow',
                        alpha=0.7),
                    fontsize=10)

ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['bottom'].set_position(('data', 0)) # x轴位置
ax.spines['left'].set_position(('data', 0)) # y轴位置

# 添加箭头 ,clip_on=False的作用是防止箭头被裁剪掉
ax.plot(1, 0, '>r',transform=ax.get_yaxis_transform(),clip_on=False) # transform=ax.get_yaxis_transform()表示箭头在y轴上
ax.plot(0, 1, '^b', transform=ax.get_xaxis_transform(),clip_on=False) # transform=ax.get_xaxis_transform()表示箭头在x轴上

ax.legend()
# 文件保存
script_name = os.path.splitext(os.path.basename(__file__))[0]
plt.savefig("figures/section3/" + script_name + '.png')
plt.show()
```

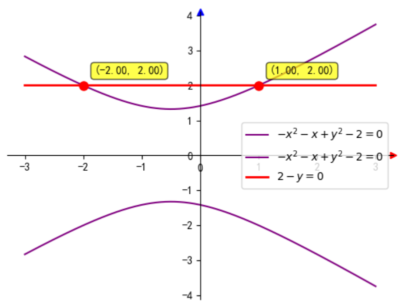

图 3.6: 圆锥曲线和直线

§4、 sympy 库

sympy 是 python 的一个库，可以帮忙带着符号进行运算. 下面用几个例子来说明.

1、 sympy 解方程

```
import sympy as sp
import matplotlib.pyplot as plt
import os
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
x = sp.symbols("x")
expr = x**2 + 4*x - 1
eq = sp.Eq(expr,0)

solution = sp.solve(eq,x)
print(solution)

fig,ax = plt.subplots()
i = 0
for s in solution:
    ax.text(0.4,0.6,f'方程${sp.latex(eq)}$的解:',fontsize = 12)
    ax.text(0.5,0.5 - 0.2 * i,f'$x_{i + 1} = {sp.latex(s)}$',fontsize = 12)
    i += 1
# 文件保存
script_name = os.path.splitext(os.path.basename(__file__))[0]
plt.savefig("figures/section4/" + script_name + '.png')
plt.show()
```

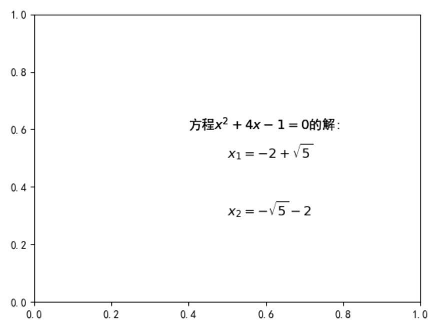

图 4.1: 解方程

2、 sympy 解二元方程

```
import sympy as sp
import matplotlib.pyplot as plt
import os
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
x,y = sp.symbols("x y")
expr1 = x**2 + 4*x - y - 1
expr2 = x + y + 1
eq1 = sp.Eq(expr1,0)
eq2 = sp.Eq(expr2,0)

solution = sp.solve([eq1,eq2],[x,y])
# print(solution)

fig,ax = plt.subplots()
i = 0
for s in solution:
    ax.text(0.2,0.6,f'方程${sp.latex(eq1)}$和${sp.latex(eq2)}$的解:',fontsize = 12)
    ax.text(0.4,0.5 - 0.1 * i,f'$x_{i + 1} = {sp.latex(s[0])}$和$y_{i + 1} = {sp.latex(s[1])}$',fontsize = 12)
    i += 1
    pass
# 文件保存
script_name = os.path.splitext(os.path.basename(__file__))[0]
plt.savefig("figures/section4/" + script_name + '.png')
plt.show()
```

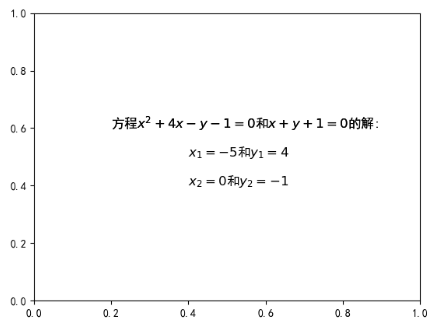

图 4.2: 解方程

3、 含参方程

例如有一个方程,ax + b = 0, 求解出 x 的表达式:

$$x = \frac{-b}{a}$$

利用 sympy 也可以方便的求解.

```
import sympy as sp
import matplotlib.pyplot as plt
import os

x,y,a,b = sp.symbols("x y a b")
expr = a*x + b - y
eq = sp.Eq(expr,0)

solution = sp.solve(eq,x)
print(solution)

result = eq.subs({a:1,b:2,y:0})
print(result)
solution = sp.solve(result,x)
print(solution)
```

一元二次方程 $ax^2+bx+c=0$ 也可以求解

```
import sympy as sp
import matplotlib.pyplot as plt
import os

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
```

```python
fig,ax = plt.subplots()

x,y = sp.symbols("x y",real = True)
a,b,c = sp.symbols("a b c",real = True)
expr = a*x**2 + b*x + c - y
eq = sp.Eq(expr,0)
eq = eq.subs({y:0})
solutions = sp.solve(eq,x)
print(solutions)

i = 0
for solution in solutions:
    plt.text(0.5,0.5 - 0.2*i,f"$x_{i + 1} = {sp.latex(solution)}$",fontsize = 12)
    i += 1
    pass
plt.text(0.5,0.8,"$ax^2 + bx + c = 0$解",fontsize = 15)
# 文件保存
script_name = os.path.splitext(os.path.basename(__file__))[0]
plt.savefig("figures/section4/" + script_name + '.png')
plt.show()
```

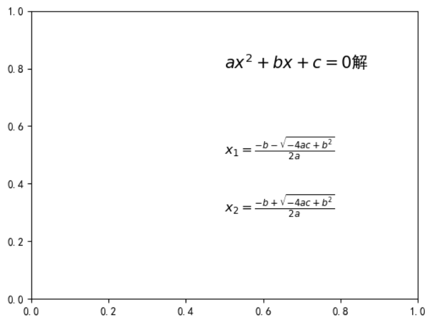

图 4.3: 含参方程

参考文献

[1] Matplotlib 官方文档, `https://matplotlib.org/stable/contents.html`

[2] NumPy 官方文档, `https://numpy.org/doc/`