12 3D Visualizations
三维可视化

三维直角坐标系中的散点、线图、网格面、等高线

善良一点，因为你遇到的每个人都在打一场更艰苦的战斗。

Be kind, for everyone you meet is fighting a harder battle.

—— 柏拉图 (Plato) | 古希腊哲学家 | 424/423 ~ 348/347 BC

- `Axes3D.plot_surface()` 绘制三维曲面
- `matplotlib.pyplot.contour()` 绘制等高线图，轴对象可以为三维
- `matplotlib.pyplot.contourf()` 绘制平面填充等高线，轴对象可以为三维
- `numpy.cumsum()` 计算给定数组中元素的累积和，返回一个具有相同形状的数组
- `numpy.exp()` 计算给定数组中每个元素的 e 的指数值
- `numpy.linspace()` 在指定的范围内创建等间隔的一维数组
- `numpy.meshgrid()` 生成多维网格化数组
- `plotly.express.data.iris()` 导入鸢尾花数据集
- `plotly.express.line()` 创建可交互的折线图的图形
- `plotly.express.scatter_3d()` 创建可交互的三维散点图
- `plotly.graph_objects.Contour()` 绘制可交互的等高线图
- `plotly.graph_objects.Scatter3d()` 绘制可交互的散点、线图
- `plotly.graph_objects.Surface()` 绘制可交互的三维曲面
- `seaborn.load_dataset()` Seaborn 库中用于加载示例数据集

12.1 三维可视化方案

本章介绍常见四种三维空间可视化方案。图 1 所示为三维直角坐标系和三个平面。

散点图 (scatter plot) 用于展示三维数据的离散点分布情况。每个数据点在三维空间中的位置由其对应的三个数值确定。通过散点图，可以观察数据点的分布、聚集程度和可能的趋势。

线图 (line plot) 可用于表示在三维空间中的曲线或路径。通过将连续的点用线段连接，可以呈现数据的演变过程或路径的形态。线图在表示运动轨迹、时间序列数据等方面很有用。

网格面图 (mesh surface plot) 展示了三维空间中表面或曲面的形状。通过将空间划分为网格，然后根据每个网格点的数值给予相应的高度或颜色，可以可视化复杂的三维数据，例如地形地貌、物理场、函数表面等。

三维等高线图 (3D contour plot) 在三维空间中绘制了等高线的曲线。这种图形通过将等高线与垂直于平面的轮廓线相结合，可以同时显示三个维度的信息。它适用于表示等值线密度、梯度分布等。

鸢尾花书《数学要素》第 6 章专门介绍三维直角坐标系。

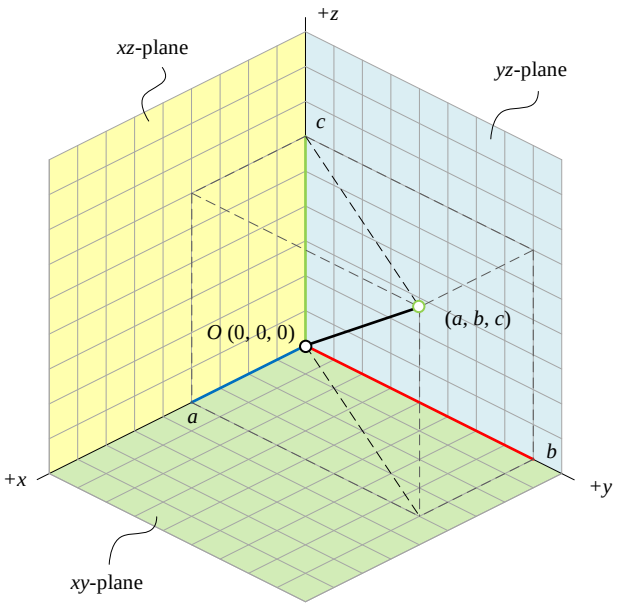

图 1. 三维直角坐标系和三个平面

三维视图视角

学过机械工程制图的同学知道，在三维空间中，我们可以将立体物体的投影投射到不同的平面上，以便更好地理解其形状和结构。图 2、图 3 展示咖啡杯在六个不同方向的投影。

以下是常见的三维立体在不同面的投影方式：

- 俯视投影 (top view) 把立体物体在垂直于其底面的平面上投影的方式。这种投影显示了物体的顶部视图，可以揭示物体在水平方向上的外形和布局。
- 侧视投影 (side view) 将立体物体在垂直于其侧面的平面上投影的方式。这种投影显示了物体的侧面视图，可以展示物体在垂直方向上的外形和结构。
- 正视投影 (front view) 把立体物体在垂直于其正面的平面上投影的方式。这种投影显示了物体的正面视图，可以展示物体在前后方向上的外形和特征。
- 斜视投影 (isometric view) 将立体物体在等角度投射到平面上的方式。它显示了物体的斜面视图，保留了物体在三个维度上的比例关系，使观察者能够同时感知物体的长度、宽度和高度。

这些不同面的投影方式可以提供不同的视角，帮助我们从多个方面理解和分析立体物体。选择合适的投影方式取决于我们关注的特定方面和目的。特别是用 Matplotlib、Plotly 绘制三维图像时，选择合适的投影方式至关重要。

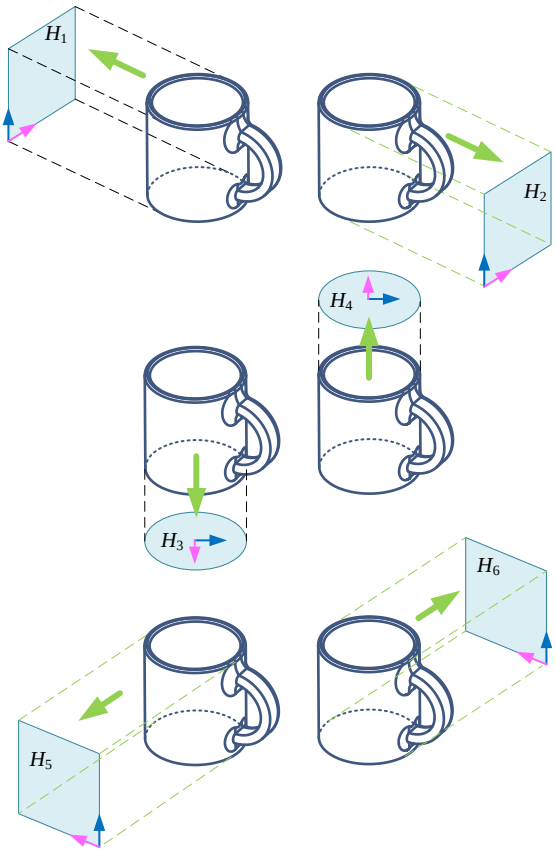

图 2. 咖啡杯六个投影方向

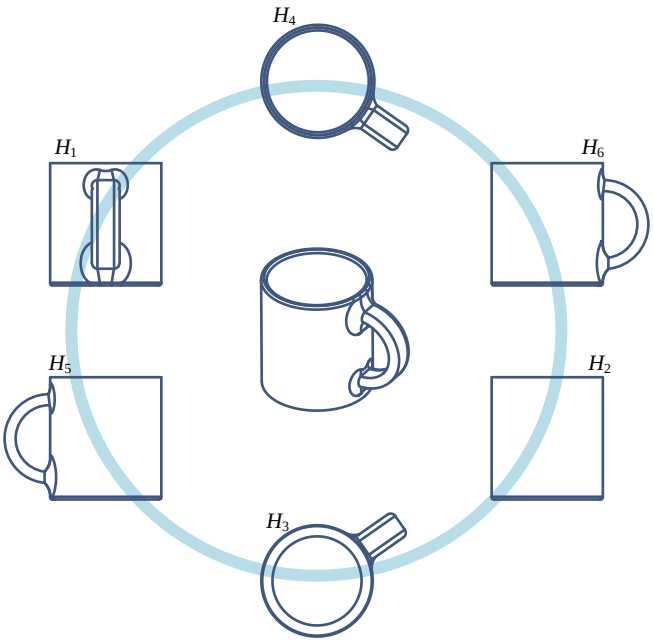

图 3. 咖啡杯在六个方向投影图像

在 Matplotlib 中，`ax.view_init(elev, azim, roll)` 方法用于设置三维坐标轴的视角，也叫相机照相位置。这个方法接受三个参数：`elev`、`azim` 和 `roll`，它们分别表示仰角、方位角和滚动角。

- 仰角 (elevation)：elev 参数定义了观察者与 *xy* 平面之间的夹角，也就是观察者与 *xy* 平面之间的旋转角度。当 elev 为正值时，观察者向上倾斜，负值则表示向下倾斜。
- 方位角 (azimuth)：azim 参数定义了观察者绕 *z* 轴旋转的角度。它决定了观察者在 *xy* 平面上的位置。azim 的角度范围是 −180 到 180 度，其中正值表示逆时针旋转，负值表示顺时针旋转。
- 滚动角 (roll)：roll 参数定义了绕观察者视线方向旋转的角度。它决定了观察者的头部倾斜程度。正值表示向右侧倾斜，负值表示向左侧倾斜。

通过调整这三个参数的值，可以改变三维图形的视角，从而获得不同的观察效果。例如，增加仰角可以改变观察者的俯视角度，增加方位角可以改变观察者在 XY 平面上的位置，增加滚动角可以改变观察者的头部倾斜程度。

类比的话，这三个角度和图 4 所示飞机的三个姿态角度类似。

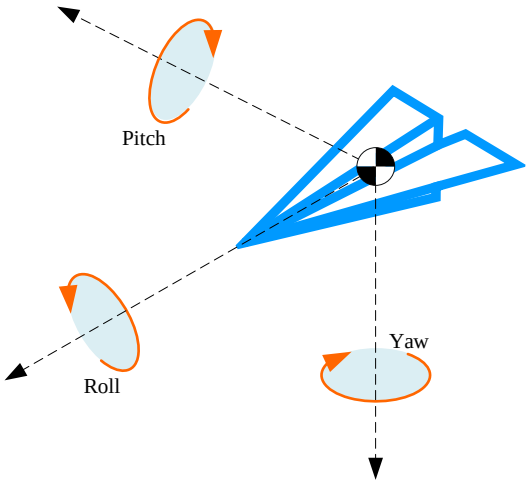

图 4. 飞机姿态的的三个角度

如图 5 所示，鸢尾花书中调整三维视图视角一般只会用 `elev`、`azim`，几乎不用使用 `roll`。

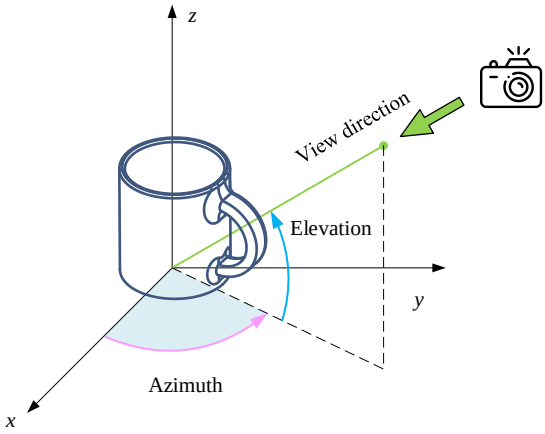

图 5. 仰角和方位角示意图

请大家在 JupyterLab 中练习如下代码，并调整仰角、方位角大小观察图像变化。

注意，ax = fig.gca(projection='3d') 已经被最新版本 Matplotlib 弃用，正确的语法为 ax = fig.add_subplot(projection='3d')。

Matplotlib设置观察视角

```python
import matplotlib.pyplot as plt
# 导入Matplotlib的绘图模块

fig = plt.figure()
# 创建一个新的图形窗口

ax = fig.add_subplot(projection='3d')
# 在图形窗口中添加一个3D坐标轴子图

ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')
# 设置坐标轴的标签

ax.set_proj_type('ortho')
# 设置投影类型为正交投影 (orthographic projection)

ax.view_init(elev=30, azim=30)
# 设置观察者的仰角为30度，方位角为30度，即改变三维图形的视角

ax.set_box_aspect([1,1,1])
# 设置三个坐标轴的比例一致，使得图形在三个方向上等比例显示

plt.show()
# 显示图形
```

图 6. 设置三维图像观察视角

有关 Matplotlib 三维视图视角，请参考：

https://matplotlib.org/stable/api/toolkits/mplot3d/view_angles.html

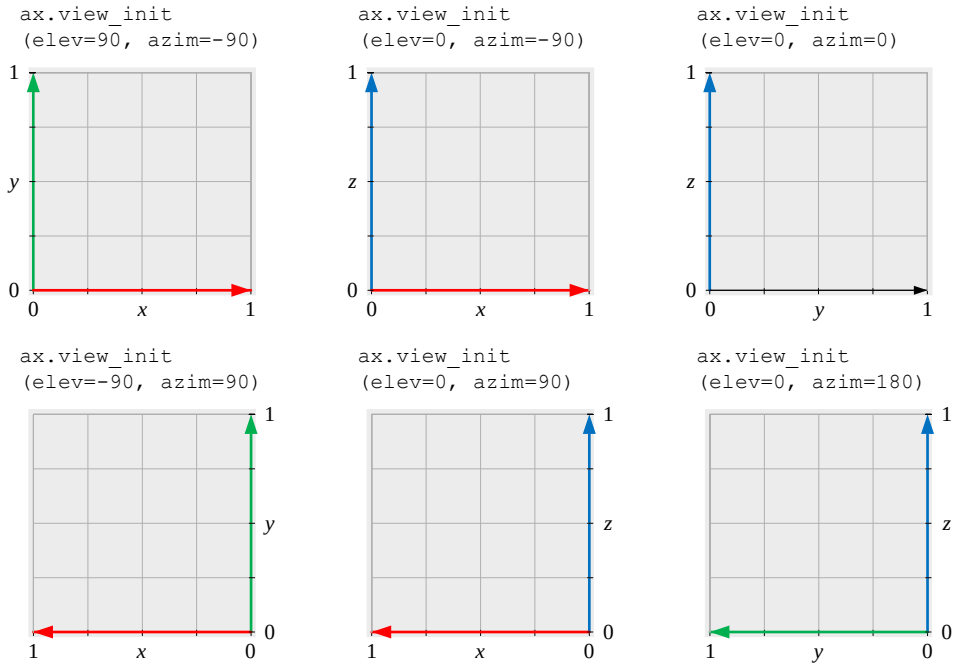

图 7. 几个特殊视角

两种投影方法

此外，大家还需要注意投影方法。上述代码采用的是正交投影。

在 Matplotlib 中，ax.set_proj_type() 方法用于设置三维坐标轴的投影类型。Matplotlib 提供了两种主要的投影类型：

- 透视投影 (perspective projection) 是默认的投影类型，如图 8 (a) 所示。简单来说就是近大远小，它模拟了人眼在观察远处物体时的视觉效果，使得远离观察者的物体显得较小。透视投影通过在观察者和图形之间创建一个虚拟的透视点，从而产生远近比例和景深感。设置方式为：ax.set_proj_type('persp')。
- 正交投影 (orthographic projection) 是另一种投影类型，如图 8 (b) 所示。它在观察者和图形之间维持固定的距离和角度，不考虑远近关系，保持了物体的形状和大小。正交投影在某些情况下可能更适合于一些几何图形的呈现，尤其是在需要准确测量物体尺寸或进行定量分析时。设置方式为：ax.set_proj_type('ortho')。

Plotly 的三维图像也是默认透视投影，想要改成正交投影对应的语法为：

```
fig.layout.scene.camera.projection.type = "orthographic"
```

图 9 展示了 3D 绘图时改变焦距对透视投影的影响。需要注意的是，Matplotlib 会校正焦距变化所带来的“缩放”效果。

透视投影中，默认焦距为 1，对应 90 度的视场角 (Field of View, FOV)。增加焦距 (1 至无穷大) 会使图像变得扁平，而减小焦距 (1 至 0 之间) 则会夸张透视效果，增加图像的视觉深度。当焦距趋近无穷大时，经过缩放校正后，会得到正交投影效果。

注意，鸢尾花书中三维图像绝大部分都是正交投影。

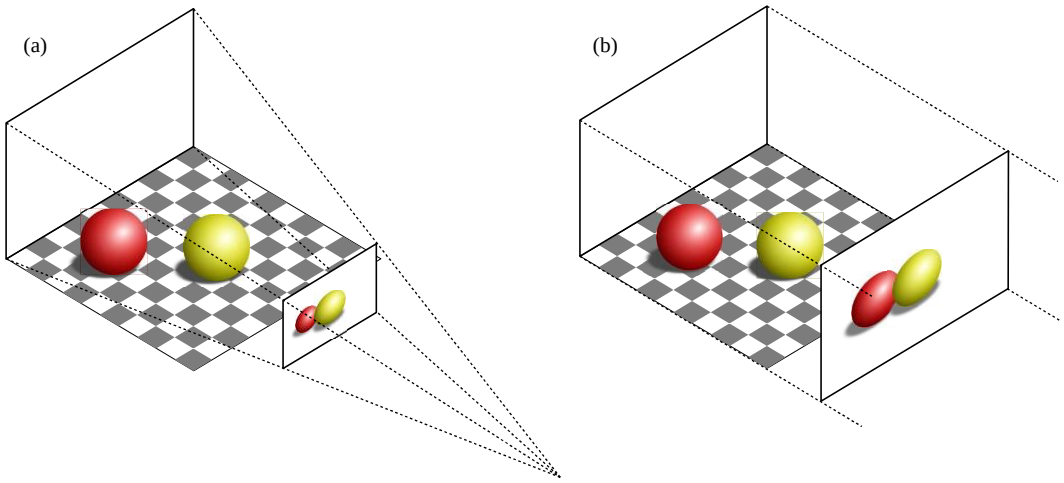

图 8. 透视投影和正交投影，来源：https://github.com/rougier/scientific-visualization-book

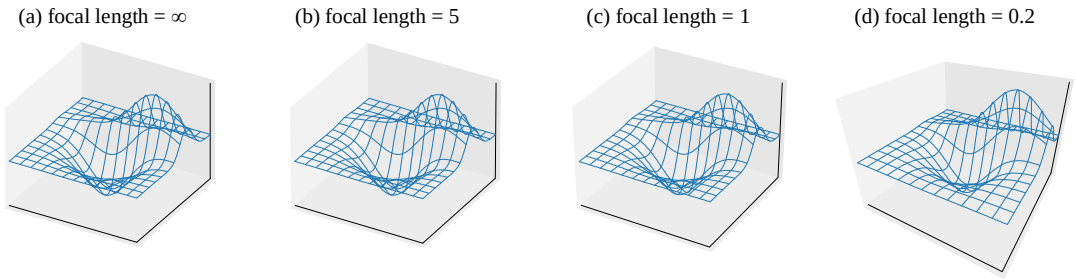

图 9. 投影焦距对结果影响；参考：https://matplotlib.org/stable/gallery/mplot3d/projections.html

12.2 散点

上一章我们利用平面散点可视化鸢尾花数据集，这一节将用三维散点图可视化这个数据集。图 10 所示为利用 Matplotlib 绘制的三维散点图，这幅图用不同颜色表征鸢尾花分类。类似图 6，请大家将图 10 投影到不同平面上。

本章配套的 Jupyter Notebook 还用 Plotly 绘制了散点图，请大家自行学习。

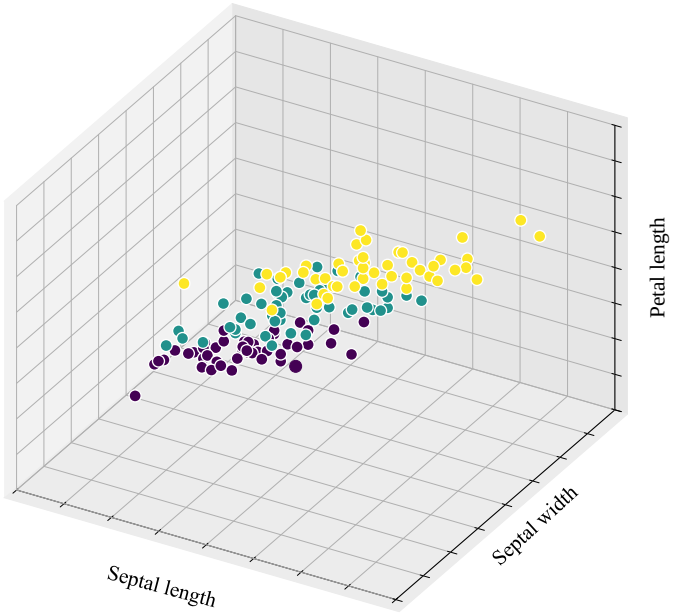

图 10. 用 Matplotlib 和绘制散点图

12.3 线图

图 11 所示为利用 Matplotlib 绘制“线图 + 散点图”可视化微粒的随机漫步。并且用散点的颜色渐进变化展示时间维度。本章配套的 Jupyter Notebook 也用 Plotly 绘制相同图像，请大家自行学习。

《数据有道》将专门介绍随机漫步。

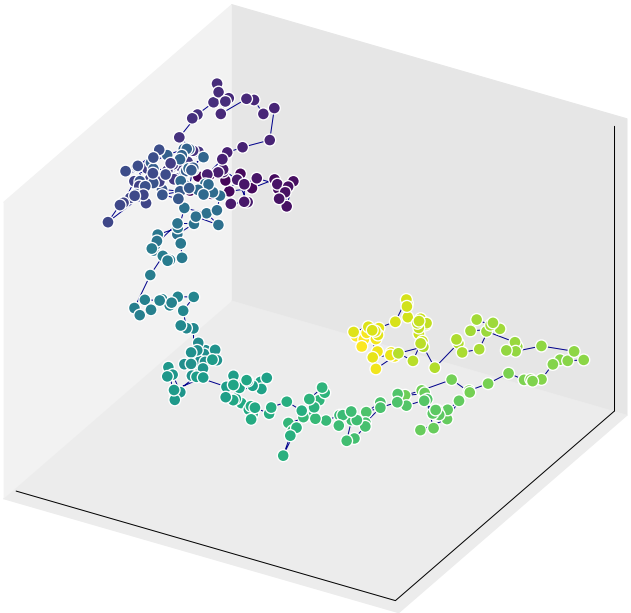

图 11. 用 Matplotlib 绘制微粒随机漫步线图

什么是随机漫步？

随机漫步是指一个粒子或者一个系统在一系列离散的时间步骤中，按照随机的方向和大小移动的过程。每个时间步骤，粒子以随机的概率向前或向后移动一个固定的步长，而且每个时间步骤之间的移动是相互独立的。随机漫步模型常用于模拟不确定性和随机性的系统，例如金融市场、扩散过程、分子运动等。通过模拟大量的随机漫步路径，可以研究粒子或系统的统计特性和概率分布。

12.4 网格面

图 12 所示为利用 Axes3D.plot_surface() 绘制的三维网格曲面。请大家思考如何在图片中加入 colorbar。本章配套的 Jupyter Notebook 也用 Plotly 绘制的三维曲面，请大家自行学习。

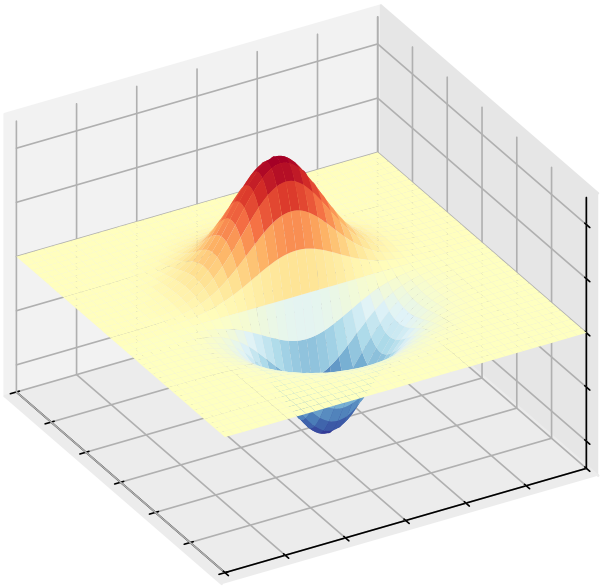

图 12. 用 Matplotlib 绘制网格曲面

12.5 三维等高线

图 13 所示为用 Matplotlib 绘制的三维等高线，这些等高线投影到水平面便得到上一章介绍的平面等高线。本章配套的 Jupyter Notebook 也用 Plotly 绘制的三维“曲面 + 等高线”，请大家自行学习。

鸢尾花书《可视之美》将介绍更多三维等高线的用法。

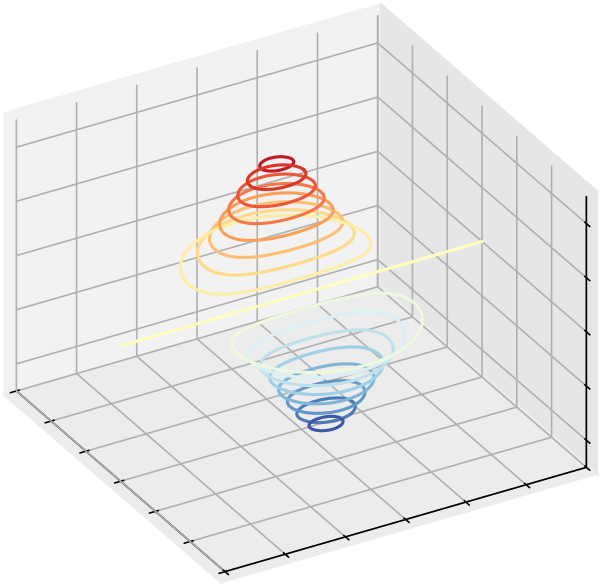

图 13. 用 Matplotlib 绘制三维等高线

请大家完成下面这道题目。

Q1. 请用分别用 Matplotlib 和 Plotly 中网格面、三维等高线可视化上一章 Q2 给出的几个二元函数。

* 本章不提供答案。