import numpy as np  
from scipy.integrate import odeint  
import matplotlib.pyplot as plt  
from temp\_2 import get\_density\_at\_depth  
from temp\_2 import read\_csv\_to\_dict  
salinity\_csv\_path = "D:\\数模\\argo\_db\\linear\_90382204\_70-28.csv"  
temperature\_csv\_path = "D:\\数模\\argo\_db\\linear\_90382204\_68-28.csv"  
salinity\_dict = read\_csv\_to\_dict(salinity\_csv\_path, 'Practical salinity adjusted - psu')  
temperature\_dict = read\_csv\_to\_dict(temperature\_csv\_path, 'Sea temperature adjusted - degree\_Celsius')  
  
  
  
def generate\_gaussian\_noise(factor=0.01):  
 return np.random.normal(0, factor)  
  
  
def model(state, t):  
 x, y, z, vx, vy, vz, \_, \_, \_ = state  
  
 # 示例参数  
 diameter = 2.7 # 圆柱体的直径，单位：米  
 height = 3.8 # 圆柱体的高度，单位：米  
 volume = 8.0 \* 2.7 \* 3.8 # 圆柱体的体积，单位：立方米  
 water\_density = get\_density\_at\_depth(z, salinity\_dict, temperature\_dict) # 海水密度，单位：kg/m^3  
 gravity = 9.8 # 重力加速度，单位：m/s^2  
 Cd = 0.47 # 阻力系数  
  
 # 计算质量  
 mass = 18500.0 # 质量，单位：kg  
  
 # 计算重力  
 weight = mass \* gravity # 注意这里是正值，因为重力方向向下  
  
 # 计算浮力  
 buoyancy = water\_density \* gravity \* volume  
  
 # 计算 z 轴方向上的加速度  
 acceleration\_z = (-buoyancy + weight) / mass  
  
 # 计算海水阻力  
 reference\_area = np.pi \* (diameter \*\* 2) / 4 # 参考面积  
  
 resistance\_z = 0.5 \* Cd \* water\_density \* reference\_area \* vz\*\*2  
  
 # 引入高斯随机漫步因子  
 noise\_factor = 0.01 # 调整噪声强度  
  
 # 计算微分方程  
 dxdt = vx + generate\_gaussian\_noise(noise\_factor)  
 dydt = vy + generate\_gaussian\_noise(noise\_factor)  
 dzdt = vz + generate\_gaussian\_noise(noise\_factor)  
 dvxdt = -resistance\_z \* np.sign(vx) / mass  
 dvydt = -resistance\_z \* np.sign(vy) / mass  
 dvzdt = acceleration\_z - resistance\_z \* np.sign(vz) / mass  
  
 return [dxdt, dydt, dzdt, dvxdt, dvydt, dvzdt, 0.0, 0.0, 0.0]  
  
# 其余部分保持不变  
def simulate\_motion(initial\_state, time\_points):  
 motion = odeint(model, initial\_state, time\_points)  
 return motion  
  
# 设置初始状态  
initial\_state = [0.0, 0.0, 0.0, 1.0, 2.0, 1.0, 0.2, 1.0, 0.2]  
  
# 设置时间点  
time\_points = np.linspace(0, 10, 100)  
  
# 模拟运动  
result = simulate\_motion(initial\_state, time\_points)  
  
# 输出末尾状态  
print("末尾状态:", result[-1])  
  
# 绘制结果  
fig = plt.figure(figsize=(10, 6), dpi=300)  
ax = fig.add\_subplot(111, projection='3d')  
  
# 绘制运动轨迹  
ax.plot(result[:, 0], result[:, 1], result[:, 2], label='Motion', linewidth=2)  
  
# 绘制位置点  
ax.scatter(result[:, 0], result[:, 1], result[:, 2], c='red', s=20, label='Position')  
  
# 设置坐标轴标签  
ax.set\_xlabel('X')  
ax.set\_ylabel('Y')  
ax.set\_zlabel('Z')  
  
# 设置标题  
ax.set\_title('Object Motion')  
  
# 显示图形  
plt.show()