**3D визуализация средствами python и matplotlib**

САМОСТОЯТЕЛЬНАЯ РАБОТА

http://www.kimrt.ru
Онлайн-курс
Суперкомпьютерные технологии в задачах моделирования

Визуализируем начальное расположение частиц в высыхающей на подложке капле. Для этого загрузите прикрепленные файлы *0.txt* и *cfg.txt*. В первом файле содержатся координаты частиц по *x*, *y* и *z* осям, во втором файле содержатся значения параметров: количество частиц, радиус капли и радиус частицы. Рассмотрим код визуализации, приведенный в листинге 1.

**Листинг 1**

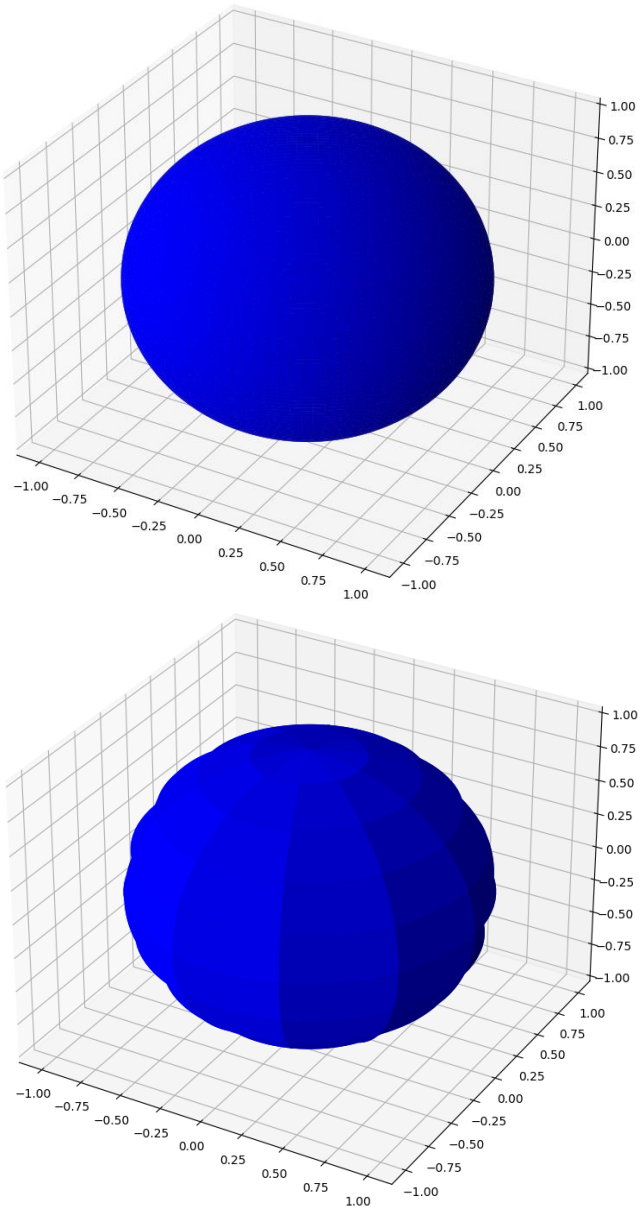
```
# импорт библиотек
import matplotlib as mpl
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
import matplotlib.pyplot as plt
import pylab
# массив частиц
arrParticles = []
# массив переменных
arrVariables = []
# индексы элементов массива с координатами
X = 0
Y = 1
Z = 2
# чтение координат частиц
```

```
with open("Data/0.txt", 'r') as f:
    for lines in f:
        l1 = lines[0:-1]
        l = l1.split(' ')
        l[X] = float(l[X])
        l[Y] = float(l[Y])
        l[Z] = float(l[Z])
        arrParticles.append(l)
    f.close()
# чтение параметров
with open("Data/cfg.txt", 'r') as f:
    for lines in f:
        l1 = lines[0:-1]
        l = l1.split('=')
        arrVariables.append(l)
    f.close()
# Идентификация параметров
for i in range(len(arrVariables)):
    if arrVariables[i][0] == 'Np':
        # количество частиц
        Np=float(arrVariables[i][1])
    if arrVariables[i][0] == 'rp':
        # радиус частицы
        rp = float(arrVariables[i][1])
    if arrVariables[i][0] == 'R':
        # радиус капли
```

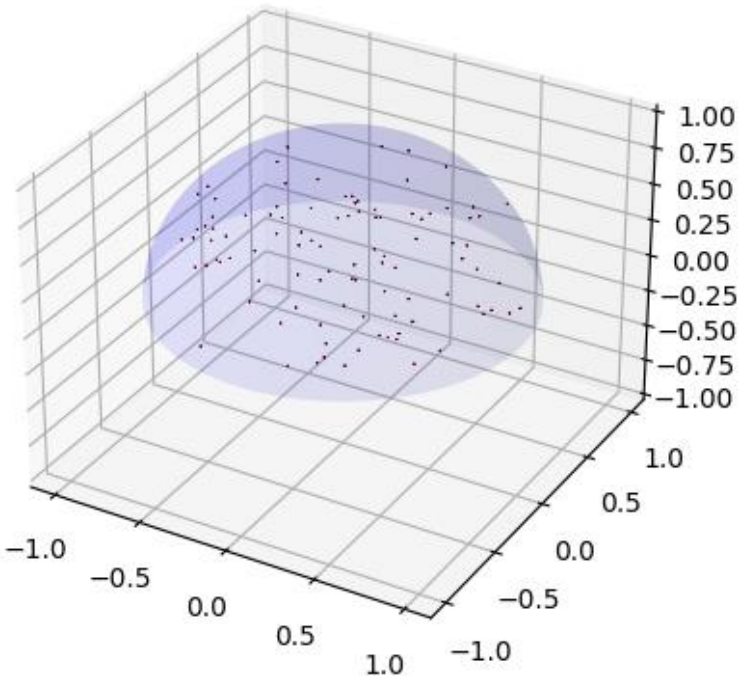
```
        R = float(arrVariables[i][1])
# создание фигуры
fig = plt.figure()
ax = fig.add_subplot(projection='3d')
# масштаб по z
ax.set_zlim(-R, R)
# рисование сферы
u = np.linspace(0, 2 * np.pi, 100)
v = np.linspace(0, np.pi/2, 100)
x = R * np.outer(np.cos(u), np.sin(v))
y = R * np.outer(np.sin(u), np.sin(v))
z = R * np.outer(np.ones(np.size(u)), np.cos(v))
ax.plot_surface(x, y, z, rstride=4, cstride=4,
color='b', alpha=0.1)
# создание массива от 0 до 2pi(100 элементов)
u = np.linspace(0, 2 * np.pi, 100)
# создание массива от 0 до pi(100 элементов)
v = np.linspace(0, np.pi, 100)
# рисование частиц
for num in range(len(arrParticles)):
    x = rp * np.outer(np.cos(u), np.sin(v)) +
arrParticles[num][X]
    y = rp * np.outer(np.sin(u), np.sin(v)) +
arrParticles[num][Y]
    z = rp * np.outer(np.ones(np.size(u)),
np.cos(v)) + arrParticles[num][Z]
```

```
    ax.plot_surface(x, y, z, rstride=4, cstride=4,
color='r', alpha=1)

# отображение рисунка

plt.show()
```

После импорта библиотек создаются списки для хранения данных о частицах. В списке *arrParticles* будут храниться координаты частиц, а список *arrVariables* необходим для временного хранения переменных. Список *arrParticles* будет двухмерным, в котором каждый вложенный список – это отдельная частица с координатами *x*, *y* и *z*. Для удобства получения значений координат сохраним их индексы расположения во вложенном списке. Далее считаем данные из файлов *0.txt* и *cfg.txt*, а также сохраним переменные из списка. Так как рассматривается частный случай, когда высота капли равна радиусу основания, используем код для построения сферы и немного модифицируем его [1]. С помощью linspace генерируем последовательности *u* и *v* от *0* до *2π* и от *0* до *π/2*. Для построения сферы необходимо генерировать последовательность *v* до π, но так как нам необходима только половина сферы, делим *π* на *2*. Далее находим произведение внешних векторов для координат с помощью outer и умножаем на радиус основания капли. С помощью *plot_surface* строим сферу. Параметры *color* и *alpha* задают цвет и прозрачность. Параметры *rstride* и *cstride* задают шаг понижения дискретизации в каждом направлении [2]. То есть эти параметры влияют на качество построения элементов. Рассмотрим примеры ниже. На первом примере значения *rstride* и *cstride* равны *1*, а на втором – *10*.

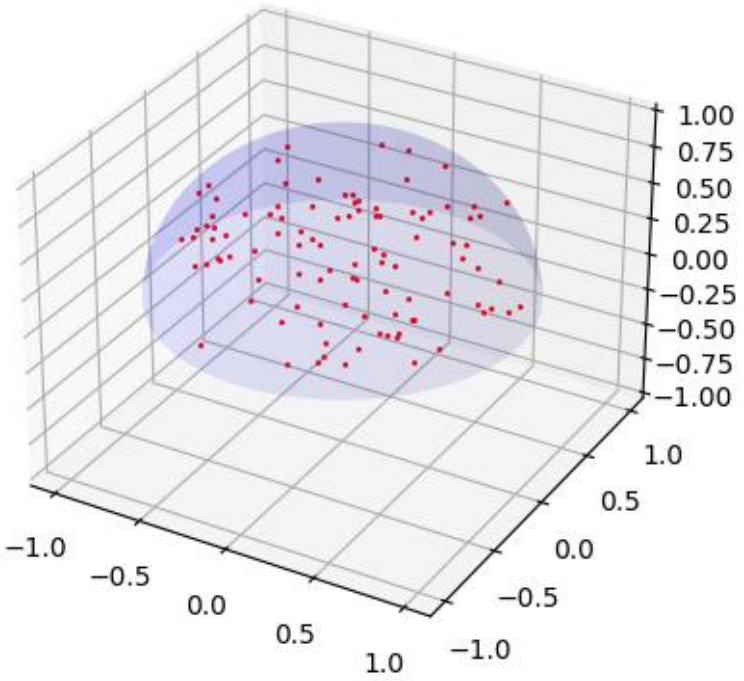



Последовательности *u* и *v* теперь задаются от *0* до *2π* и от *0* до *π*, так как теперь необходимо строить всю сферу. Далее в цикле перебираются все частицы и строятся сферы с радиусом равным радиусу частиц. Также необходимо прибавлять координаты частиц.




Еще возможно использование точек, а не сфер при визуализации. Такой способ быстрее работает, но размер точки не будет равен размеру частицы. Для использования точек, замените код в цикле на следующую строку.

*ax.plot(arrParticles[num][X], arrParticles[num][Y], arrParticles[num][Z], lw=0, color='r', label='parametric curve', marker='o', markersize=1)*



Дополнительные материалы: как создавать 3d модели с помощью Python [3], 3D моделирование в Python [4], Open3D [5].

http://www.kimrt.ru/index/course_stm/0-24

*Проект реализуется победителем Конкурса на предоставление грантов преподавателям магистратуры 2020/2021 благотворительной программы «Стипендиальная программа Владимира Потанина» Благотворительного фонда Владимира Потанина.*

**Источники**

1. Сфера. URL: https://matplotlib.org/stable/gallery/mplot3d/surface3d_2.html#sphx-glr-gallery-mplot3d-surface3d-2-py
2. Mplot3d toolkit. URL: https://matplotlib.org/3.5.0/tutorials/toolkits/mplot3d.html
3. Как создавать 3d модели с помощью Python. URL: https://habr.com/ru/post/411899/
4. 3D моделирование в Python. URL: https://habr.com/ru/post/572760/
5. Open3D. URL: http://www.open3d.org/