Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра комп'ютерного моделювання процесів і систем

ЗВІТ

з лабораторної роботи №3

"Засоби побудови полів у Matplotlib"

з курсу

«Алгоритми та моделі збору, аналізу та візуалізації даних»

Виконав: студент групи ІКМ-М222к Черкас Ю.В.

Перевірила: аспірантка Рикова В.О.

Харків 2023р

Варіант №15

1. Візуалізацію скалярного поля. Знайдіть його градієнт та візуалізуйте його як плоске векторне поле;

$$u(x,y) = 7\,ln(x^2 + \frac{1}{13}) - 4\,sin(xy)\,; x\,, y \in [-10\,; 1\,0]$$

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

n = 256
x = np.linspace(-10., 10., n) # діапазон по X
y = np.linspace(-10., 10., n) # діапазон по Y
X, Y = np.meshgrid(x, y)
Z = 7 * np.log(X ** 2 + 1/13) + 4 * np.sin(X * Y) # Формула скалярного поля

plt.title('Скалярне поле ' + r'$u(x,y)=-7\ln(x^2+1/13) + 4\sin(xy)$')
plt.pcolormesh(X, Y, Z)
plt.show()
```

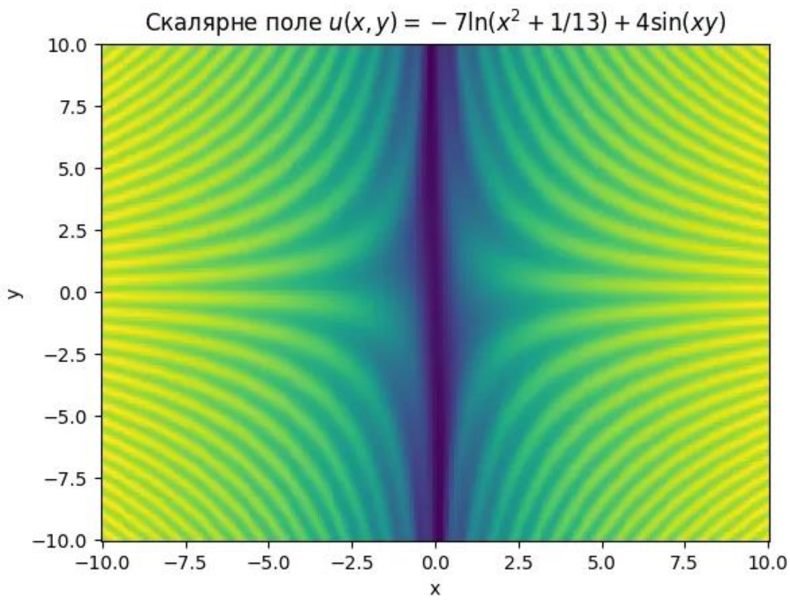

Рисунок 1 – Графік скалярного поля

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

n = 20
```

```
x = np.linspace(-10., 10., n) # діапазон по X
y = np.linspace(-10., 10., n) # діапазон по Y
X, Y = np.meshgrid(x, y)
Z = 7 * np.log(X ** 2 + 1/13) + 4 * np.sin(X * Y) # Формула скалярного поля
Z_dx, Z_dy = np.gradient(Z)

# Векторне поле
plt.quiver(X, Y, Z_dx, Z_dy)
plt.title('Векторне поле ' + r'$u(x,y)=-7\ln(x^2+1/13) + 4\sin(xy)$')
plt.show()

# Лінії потоку поля
fig, ax = plt.subplots()
ax.set_aspect('equal', 'box')
ax.streamplot(X, Y, Z_dx, Z_dy, color=Z, cmap='viridis')
plt.title('Лінії потоку поля ' + r'$u(x,y)=-7\ln(x^2+1/13) + 4\sin(xy)$')
plt.show()
```

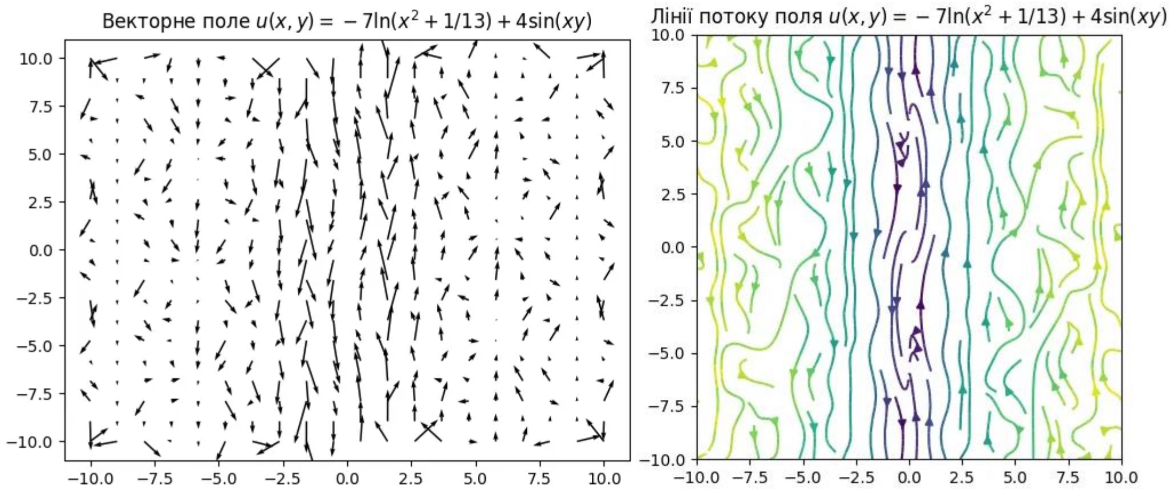

Рисунок 2 – Візуалізація векторного поля

2. Побудуйте візуалізацію плоского векторного поля як за допомогою векторів та ліній току з бібліотеки matplotlib та за допомогою коду з лістингу;

$$F = (x^2 y; -y); x, y \in [-10; 10]$$

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

def u(x, y):
    return y*x**2
```

```python
def v(x, y):
    return -y

xx, yy = np.meshgrid(np.linspace(-10, 10, 10), np.linspace(-10, 10, 10))
u_val = u(xx, yy)
v_val = v(xx, yy)

plt.quiver(xx, yy, u_val, v_val)
plt.title('Векторне поле ' + r'$F = (x^2y; -y)$')
plt.show()

fig, ax = plt.subplots()
plt.streamplot(xx, yy, u_val, v_val)
plt.title('Лінії потоку ' + r'$F = (x^2y; -y)$')
plt.show()
```

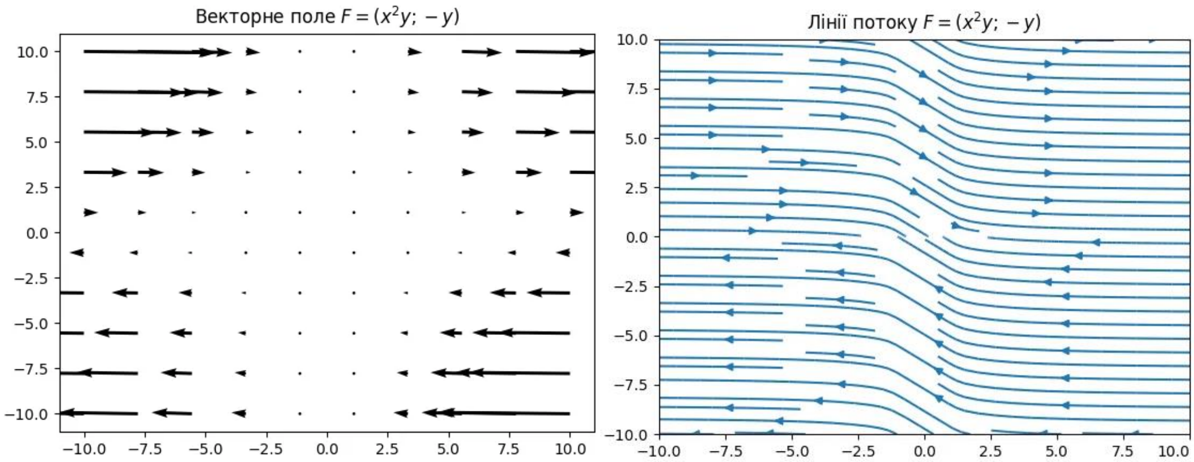

Рисунок 3 – Візуалізація векторного поля

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

def u(x, y):
    return y*x**2
def v(x, y):
    return -y

def create_stream_line(x0, y0, u, v, t0=0, t1=0.001, dt=0.0001):
    t = np.arange(t0, t1, dt)
    xx_new = np.zeros_like(t)
    yy_new = np.zeros_like(t)
    xx_new[0] = x0
    yy_new[0] = y0

    for i in range(1, t.size):
```

```
        xx_new[i] = x0 + u(x0, y0) * dt
        yy_new[i] = y0 + v(x0, y0) * dt
        x0, y0 = xx_new[i], yy_new[i]
    return xx_new, yy_new

for i in range(-10, 10):
    for j in range(-10, 10):
        x1, y1 = create_stream_line(i, j, u, v)
        plt.plot(x1, y1)

plt.title('Лінії току ' + r'$F = (x^2y; -y)$')
plt.show()
```

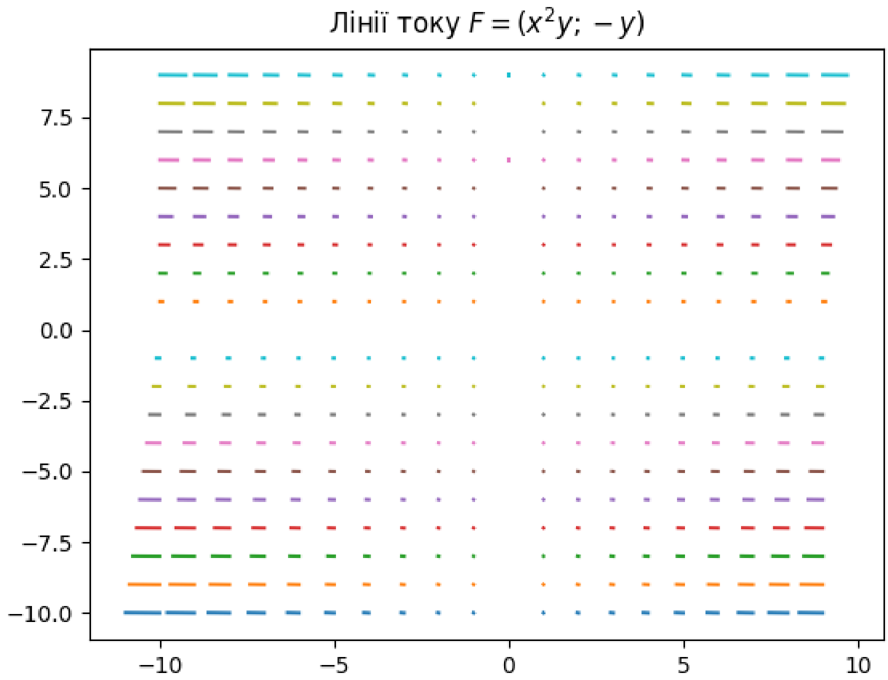

Рисунок 4 – Візуалізація ліній току

3. Побудуйте тривимірну візуалізацію векторного поля; За додатковий бал (не обов'язково) модернізуйте алгоритм побудови ліній току на випадок 3-вимірного поля.

$$F = \left(\frac{x+z}{x^2}; \frac{1}{y}; \frac{1}{z}\right); x, y, z \in [-10; 10]$$

```
from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt
import numpy as np

x, y, z = np.meshgrid(np.arange(-10, 10, 2.5),
```

```
                      np.arange(-10, 10, 2.5),
                      np.arange(-10, 10, 2.5))

u = (x + z)/x**2
v = 1/y
w = 1/z

ax = plt.figure().add_subplot(projection='3d')
ax.quiver(x, y, z, u, v, w, length=2, color = 'black')
plt.title(r'$F=(\frac{x+z}{x^2}; \frac{1}{y}; \frac{1}{z})$')
plt.show()
```

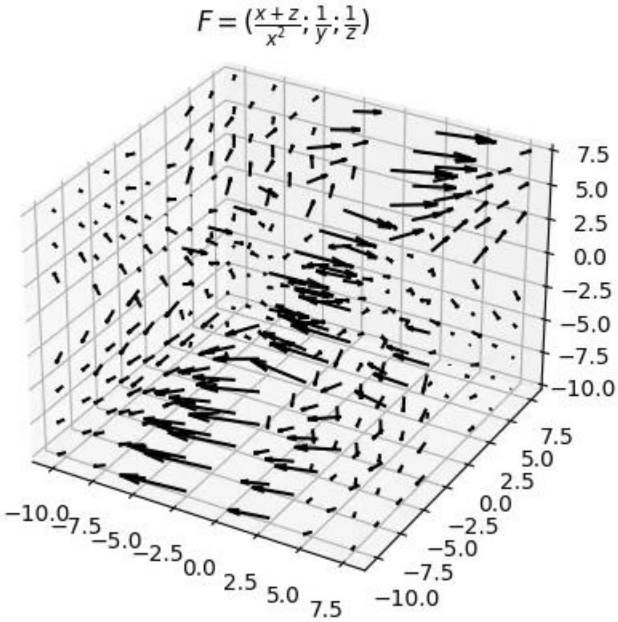

Рисунок 5 – Тривимірна візуалізація векторного поля

4. Побудуйте візуалізацію тензорного поля за допомогою еліпсоїдів, кубоїдів, циліндрів та будь-якого суперквадру.

$$T = \begin{pmatrix} \frac{ln(x)}{sin(x)} & \sqrt{x}/y & \sqrt{y}/z \\ & \frac{ln(y)}{sin(y)} & \sqrt{z}/x \\ & & \frac{ln(z)}{sin(z)} \end{pmatrix}, x, y, z \in [1..3]$$

```python
import numpy as np
import matplotlib.pyplot as plt
from mayavi import mlab
import glyph_visualization_lib as gvl

def main():
    x = np.linspace(1, 3, 8)
    y = np.linspace(1, 3, 8)
    z = np.linspace(1, 3, 8)
    X, Y, Z = np.meshgrid(x, y, z)

    stress_tensor = np.array([
        [np.log(X)/np.sin(X),   np.sqrt(X)/Y,           np.sqrt(Y)/Z],
        [np.sqrt(X)/Y,          np.log(Y)/np.sin(Y),    np.sqrt(Z)/X],
        [np.sqrt(Y)/Z,          np.sqrt(Z)/X,           np.log(Z)/np.sin(Z)]
    ])

    vm_stress = gvl.get_von_Mises_stress(stress_tensor)
    glyph_radius = 0.1
    limits = [np.min(vm_stress), np.max(vm_stress)]
    colormap = plt.get_cmap('rainbow', 120)
    fig = mlab.figure(bgcolor=(1, 1, 1))
    fig2 = plt.figure()
    ax = fig2.add_subplot(111, projection='3d')

    for i in range(x.size):
        for j in range(y.size):
            for k in range(z.size):
                center = [x[i], y[j], z[k]]
                data = stress_tensor[:, :, i, j, k]
                color = colormap(gvl.get_colormap_ratio_on_stress(vm_stress[i,
j, k], limits))[:3]

                x_g, y_g, z_g = gvl.get_glyph_data(center, data, limits,
glyph_points=12, glyph_radius=glyph_radius,
                                                   glyph_type=2,
                                                   superquadrics_option=2)
                mlab.mesh(x_g, y_g, z_g, color=color)
    mlab.move(forward=1.8)
    mlab.savefig("superquadric-Kindlmann_modified-viz.png", size=(100, 100))
    mlab.show()

    pass


if __name__ == '__main__':
    main()
```

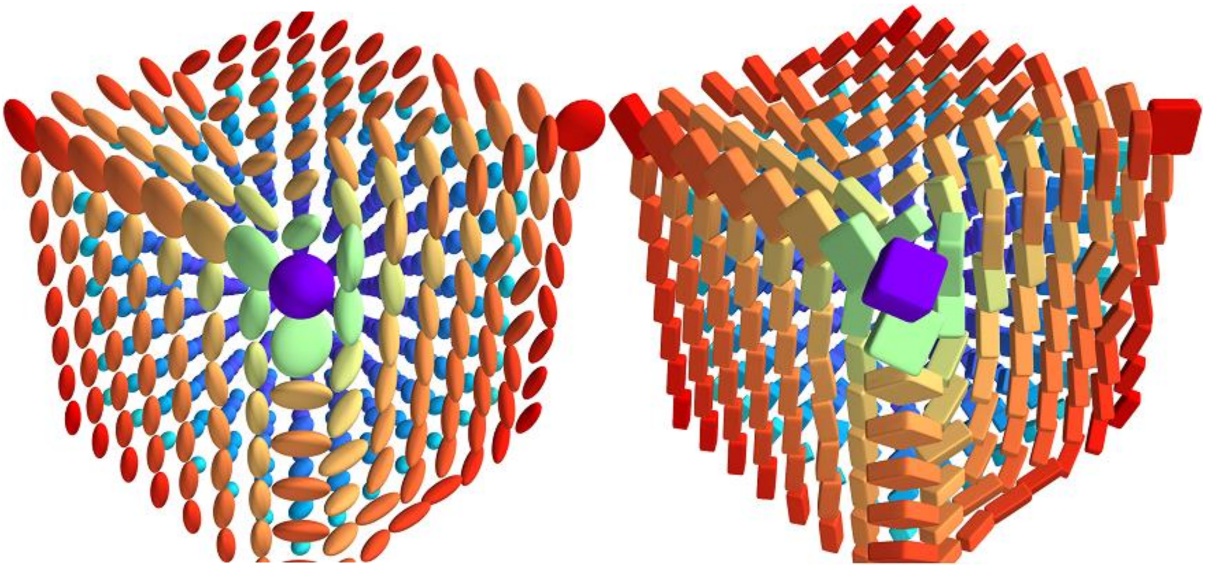

Рисунок 6 – Візуалізація тензорного поля еліпсоїдами та кубоїдами

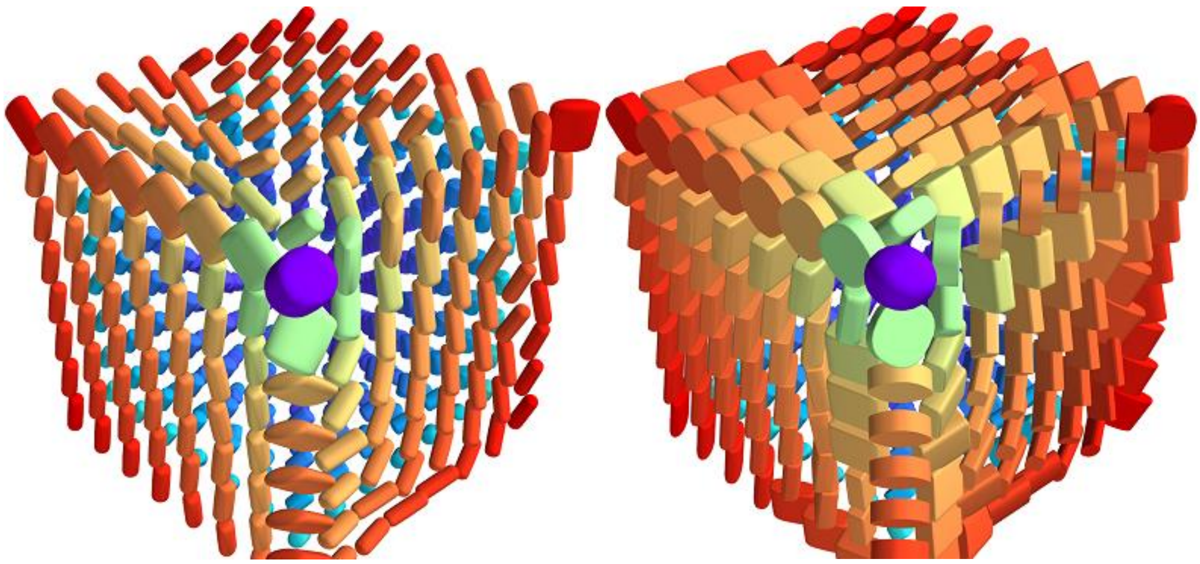

Рисунок 7 – Візуалізація тензорного поля циліндрами та суперквадратами Кінделмана

Висновок

На даній лабораторній роботі ми дослідили можливості бібліотеки Matplotlib для мови програмування Python при візуалізації полів. Здобули навики візуалізації скалярних та векторних полів в 2D та 3D розмірностях. Оцінили відмінності візуалізації градієнтів за допомогою векторних полів та ліній потоку. Набули навики візуалізації тензорних полів гліфами за допомогою спеціалізованих бібліотек glyph_visualization_lib та mayavi.