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;

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()

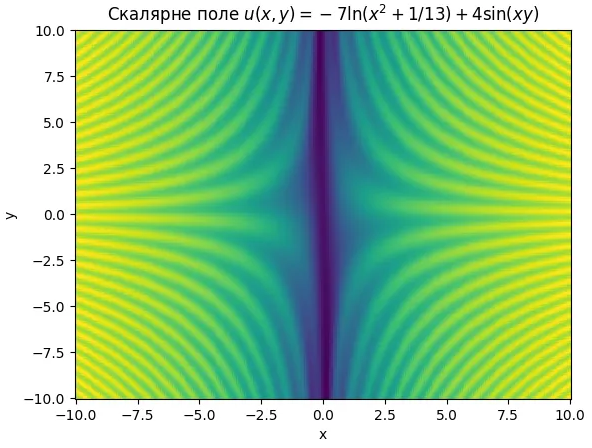


Рисунок 1 – Графік скалярного поля

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()

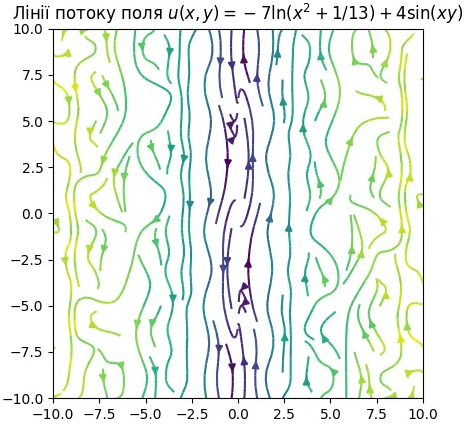
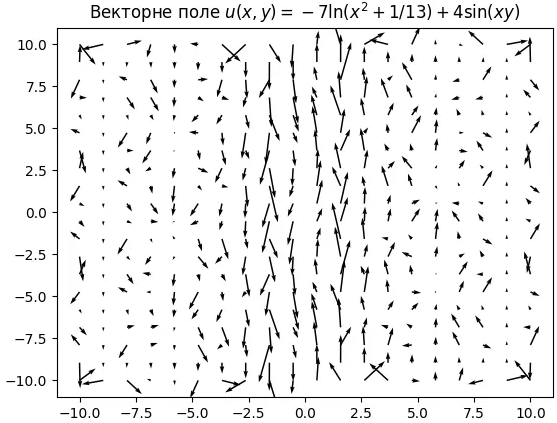


Рисунок 2 – Візуалізація векторного поля

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

import numpy as np

import matplotlib.pyplot as plt

def u(x, y):

    return y\*x\*\*2

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()

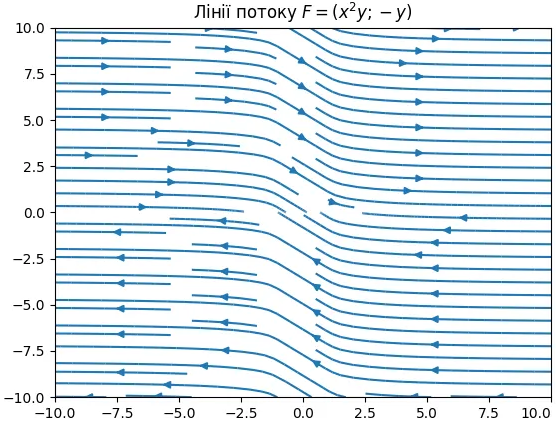
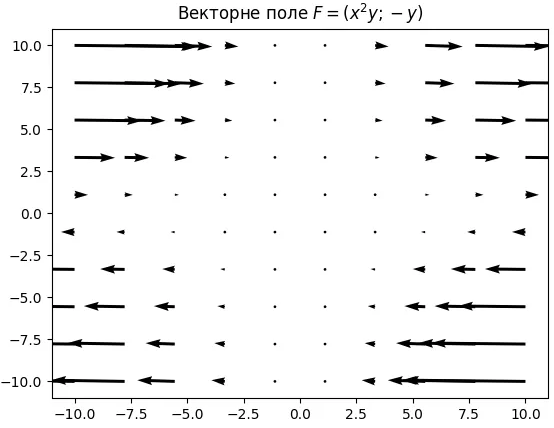


Рисунок 3 – Візуалізація векторного поля

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

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()

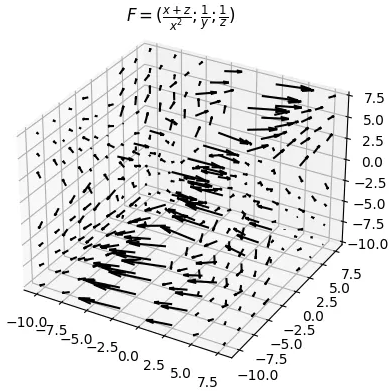


Рисунок 4 – Тривимірна візуалізація векторного поля

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

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()

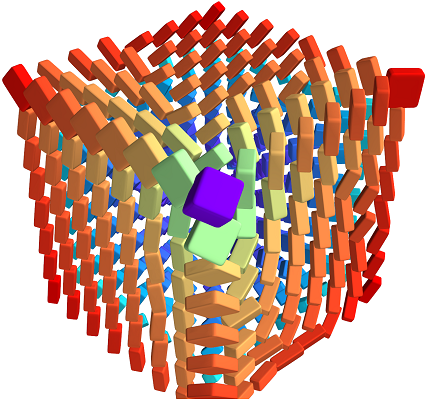
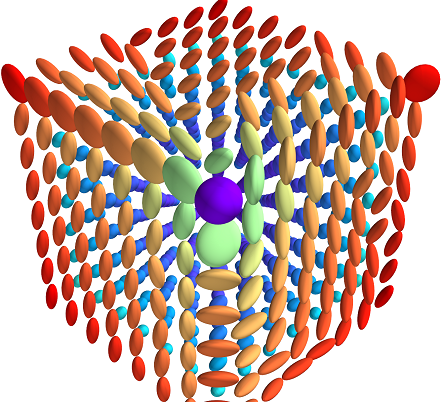


Рисунок 5 – Візуалізація тензорного поля еліпсоїдами та кубоїдами

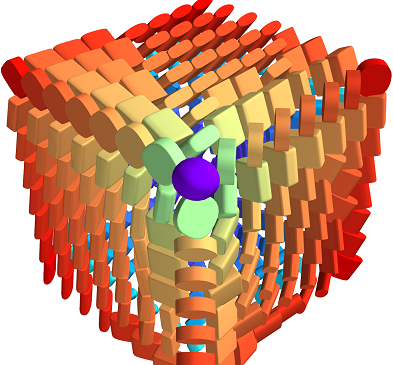
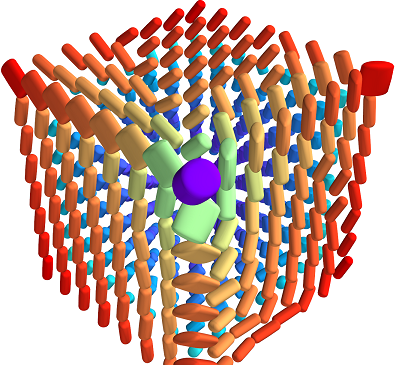
****

Рисунок 6 – 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.