**Лабораторна робота №2**

Малювання тензорних полів за допомогою Matplotlib

1. Візуалізувати скалярне поле. Знайти його градієнт та візуалізувати його у вигляді плоского векторного поля. Варіант 2.



Рішення:

import numpy as np

import matplotlib.pyplot as plt

# Задання області

x = np.linspace(0, 5, 50)

y = np.linspace(0, 5, 50)

X, Y = np.meshgrid(x, y)

# Скалярне поле

Z = X \* np.sqrt(Y) + Y \* np.sqrt(X)

# Візуалізація скалярного поля

plt.figure(figsize=(10, 6))

plt.contourf(X, Y, Z, cmap='viridis')

plt.colorbar(label='u(x,y)')

plt.title('Скалярне поле u(x,y)')

plt.xlabel('x')

plt.ylabel('y')

plt.show()

# Обчислення градієнту

grad\_x = np.sqrt(Y) + 0.5 \* X / np.sqrt(Y)

grad\_y = np.sqrt(X) + 0.5 \* Y / np.sqrt(X)

# Візуалізація плоского векторного поля

plt.figure(figsize=(10, 6))

plt.quiver(X, Y, grad\_x, grad\_y, scale=20)

plt.title('Градієнт скалярного поля')

plt.xlabel('x')

plt.ylabel('y')

plt.show()

Зображення, що містить знімок екрана, текст, Барвистість, Графік

Автоматично згенерований опис

Зображення, що містить знімок екрана, текст, Прямокутник, комп’ютер

Автоматично згенерований опис

1. Візуалізувати плоске векторне поле як за допомогою векторів, так і за допомогою ліній струму з бібліотеки matplotlib та за допомогою коду з лістингу.

Зображення, що містить Шрифт, типографія, каліграфія, текст

Автоматично згенерований опис

Рішення:

import numpy as np

import matplotlib.pyplot as plt

def u(x, y):

return x \*\* 2 + 2 \* y

def v(x, y):

return y \*\* 2 + 2 \* x

xx, yy = np.meshgrid(np.linspace(-4, 4, 10),

np.linspace(-4, 4, 10))

u\_val = u(xx, yy)

v\_val = v(xx, yy)

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)

plt.quiver(xx, yy, u\_val, v\_val)

plt.title('Векторне поле')

plt.subplot(1, 2, 2)

plt.streamplot(xx, yy, u\_val, v\_val)

plt.title('Лінії току')

plt.tight\_layout()

plt.show()

Зображення, що містить візерунок, ряд, дизайн

Автоматично згенерований опис

1. Побудувати тривимірну візуалізацію векторного поля;

Зображення, що містить текст, ряд, Шрифт, почерк

Автоматично згенерований опис

Рішення:

from mpl\_toolkits.mplot3d import Axes3D

import matplotlib.pyplot as plt

import numpy as np

# Функція для обчислення векторного поля

def vector\_field(x, y, z):

denominator = x\*\*2 + y\*\*2 + z\*\*2

F1 = y \* z / denominator

F2 = x \* z / denominator

F3 = x \* y / denominator

return F1, F2, F3

# Сітка точок

x = np.linspace(-3, 4, 10)

y = np.linspace(-3, 4, 10)

z = np.linspace(-3, 4, 10)

X, Y, Z = np.meshgrid(x, y, z)

# Обчислення компоненти векторного поля на сітці точок

U, V, W = vector\_field(X, Y, Z)

# Візуалізація векторного поля

fig = plt.figure()

ax = fig.add\_subplot(projection='3d')

# Відображення векторного поля за допомогою quiver

ax.quiver(X, Y, Z, U, V, W, length=0.2, color='black')

# Налаштування відображення

ax.set\_xlabel('X axis')

ax.set\_ylabel('Y axis')

ax.set\_zlabel('Z axis')

ax.set\_title('3D Vector Field Visualization')

plt.show()

Зображення, що містить ескіз, візерунок, чорно-білий, схема

Автоматично згенерований опис

1. Побудувати візуалізацію тензорного поля, використовуючи еліпсоїди, кубоїди, циліндри та будь-який суперквадрант.

Зображення, що містить текст, Шрифт, почерк, ряд

Автоматично згенерований опис

Рішення:

import numpy as np

import matplotlib.pyplot as plt

from mayavi import mlab

import glyph\_visualization\_lib as gvl

def visualize\_tensor\_field(glyph\_type, superquadrics\_option=0, glyph\_name=''):

x = np.linspace(-2 \* np.pi, 2 \* np.pi, 8, dtype=float, endpoint=True)

y = np.linspace(-2 \* np.pi, 2 \* np.pi, 8, dtype=float, endpoint=True)

z = np.linspace(-2 \* np.pi, 2 \* np.pi, 8, dtype=float, endpoint=True)

X, Y, Z = np.meshgrid(x, y, z)

tensor\_field = np.zeros((3, 3, X.shape[0], X.shape[1], X.shape[2]))

tensor\_field[0, 0] = np.sin(X)

tensor\_field[0, 1] = X + Y

tensor\_field[0, 2] = X + Z

tensor\_field[1, 1] = np.cos(Y)

tensor\_field[1, 2] = Y + Z

tensor\_field[2, 2] = np.cos(Z)

vm\_stress = gvl.get\_von\_Mises\_stress(tensor\_field)

glyph\_radius = 0.25

limits = [np.min(vm\_stress), np.max(vm\_stress)]

colormap = plt.get\_cmap('rainbow', 120)

fig = mlab.figure(bgcolor=(1, 1, 1))

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 = tensor\_field[:, :, 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=glyph\_type, superquadrics\_option=superquadrics\_option)

mlab.mesh(x\_g, y\_g, z\_g, color=color)

mlab.move(forward=1.8)

filename = f"tensor\_field\_visualization\_{glyph\_name}.png"

mlab.savefig(filename, size=(1000, 1000))

mlab.show()

if \_\_name\_\_ == '\_\_main\_\_':

# Visualize using ellipsoids

visualize\_tensor\_field(glyph\_type=2, glyph\_name='ellipsoids')

# Visualize using cuboids

visualize\_tensor\_field(glyph\_type=0, glyph\_name='cuboids')

# Visualize using cylinders

visualize\_tensor\_field(glyph\_type=1, glyph\_name='cylinders')

# Visualize using superquadrics

visualize\_tensor\_field(glyph\_type=3, superquadrics\_option=2, glyph\_name='superquadrics')

Зображення, що містить Барвистість, коло, Симетрія, дизайн

Автоматично згенерований опис

Рисунок 1 – Кубоїди

Зображення, що містить Барвистість, коло, Симетрія, дизайн

Автоматично згенерований опис

Рисунок 2 – Еліпсоїди

Зображення, що містить Барвистість, коло, Симетрія, дизайн

Автоматично згенерований опис

Рисунок 3 – Циліндри

Зображення, що містить Барвистість, дизайн

Автоматично згенерований опис

Рисунок 4 – Суперквадрант

Виконала: Павленко Дарина

Група: ІКМ-М223в