**Лабораторна робота №3**

Засоби побудови полів у Matplotlib

1. Побудувати візуалізацію скалярного поля. Знайти його градієнт та візуалізувати його як плоске векторне поле. Варіант 2.



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()

Задано область за допомогою np.linspace та np.meshgrid. Створено скалярне поле Z за допомогою заданої формули. Побудовано візуалізацію скалярного поля за допомогою plt.contourf. Обчислено компоненти градієнту grad\_x та grad\_y за допомогою формули для часткових похідних. Побудовано візуалізацію плоского векторного поля за допомогою plt.quiver.

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

Автоматично згенерований опис

Зображення, що містить знімок екрана, текст, Прямокутник

Автоматично згенерований опис

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 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()

Спочатку, ми визначаємо функції u(x, y) та v(x, y), що задають компоненти векторів поля у відповідних напрямках. Потім генеруємо координатну сітку у площині за допомогою np.meshgrid, яка представляє собою масиви xx та yy координат точок сітки у двох напрямках.

За допомогою функцій u(xx, yy) та v(xx, yy) обчислюємо значення компонентів векторів поля на всій сітці. Далі, використовуємо plt.quiver для побудови векторів поля. Ця функція відображає вектори з початками в точках сітки та напрямленими відповідно до значень u\_val та v\_val. Використовуємо plt.streamplot для побудови ліній току, що показують напрямок та інтенсивність руху у векторному полі.

Зображення, що містить візерунок, ряд, схема, знімок екрана

Автоматично згенерований опис

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



import numpy as np

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

# Обчислення компонент векторного поля

def F(x, y, z):

r\_squared = x\*\*2 + y\*\*2 + z\*\*2

return y\*z/r\_squared, x\*z/r\_squared, x\*y/r\_squared

# Генерування значень для x, y, z

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

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

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

xx, yy, zz = np.meshgrid(x, y, z, indexing='ij')

# Обчислення компонент векторного поля на генерованій сітці

u, v, w = F(xx, yy, zz)

# Побудова візуалізації

fig = plt.figure(figsize=(10, 8))

ax = fig.add\_subplot(111, projection='3d')

# Побудова векторного поля

ax.quiver(xx, yy, zz, u, v, w, length=0.2, normalize=True, color='b')

# Модернізований алгоритм побудови ліній току

start\_points = np.stack([np.random.choice(coord.flatten(), 10) for coord in [xx, yy, zz]], axis=-1)

stream = ax.streamplot(xx[:,0,0], yy[0,:,0], zz[0,0,:], u, v, w, 2, start\_points=start\_points, color='r')

# Налаштування відображення

ax.set\_xlabel('X')

ax.set\_ylabel('Y')

ax.set\_zlabel('Z')

ax.set\_title('Тривимірне векторне поле та лінії току')

plt.show()

Функція F(x, y, z) обчислює компоненти векторного поля відповідно до заданої формули, де кожна компонента залежить від координат x, y, та z. Генеруються значення x, y, та z в заданих межах. За допомогою np.meshgrid генерується тривимірна координатна сітка xx, yy, та zz, а за допомогою функції F, обчислюються значення компонент векторного поля u, v, та w на всій сітці. Створюється новий об'єкт Figure та відповідний Axes3D для побудови тривимірної графіки. За допомогою ax.quiver відображаються вектори поля з початками в точках сітки та напрямленими відповідно до обчислених компонент.

Для алгоритму використовується ax.streamplot для побудови ліній току. Додано можливість вибору початкових точок start\_points, які випадково обираються серед координат сітки.

Зображення, що містить візерунок, ескіз, дизайн

Автоматично згенерований опис

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

Зображення, що містить текст, Шрифт, почерк, ряд

Автоматично згенерований опис

import numpy as np

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

# Обчислюємо значення тензорного поля

x\_values = np.linspace(-2\*np.pi, 2\*np.pi, 100)

y\_values = np.linspace(-2\*np.pi, 2\*np.pi, 100)

z\_values = np.linspace(-2\*np.pi, 2\*np.pi, 100)

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

# Задання тензорного поля

tensor\_field = np.array([

[np.sin(X), X + Y, X + Z],

[X+Y, np.cos(Y), Y + Z],

[X+Y, Y+Z, np.cos(Z)]

])

# Функція для побудови тензорного поля з різними геометричними фігурами

def plot\_tensor\_field(geometry, tensor\_field):

fig = plt.figure()

ax = fig.add\_subplot(111, projection='3d')

for i in range(tensor\_field.shape[0]):

for j in range(tensor\_field.shape[1]):

for k in range(tensor\_field.shape[2]):

if not np.all(np.isnan(tensor\_field[i, j, k])):

if geometry == 'ellipsoid':

ax.scatter(X[i, j, k], Y[i, j, k], Z[i, j, k], color='b', alpha=0.5)

elif geometry == 'cuboid':

ax.scatter(X[i, j, k], Y[i, j, k], Z[i, j, k], color='r', alpha=0.5, marker='s')

elif geometry == 'cylinder':

ax.scatter(X[i, j, k], Y[i, j, k], Z[i, j, k], color='g', alpha=0.5, marker='|')

elif geometry == 'superquadric':

ax.scatter(X[i, j, k], Y[i, j, k], Z[i, j, k], color='m', alpha=0.5, marker='^')

ax.set\_xlabel('X')

ax.set\_ylabel('Y')

ax.set\_zlabel('Z')

ax.set\_title(f'Tensor Field with {geometry}s')

# Побудова тензорного поля з різними геометричними фігурами

plot\_tensor\_field('ellipsoid', tensor\_field)

plot\_tensor\_field('cuboid', tensor\_field)

plot\_tensor\_field('cylinder', tensor\_field)

plot\_tensor\_field('superquadric', tensor\_field)

plt.show()

За допомогою np.meshgrid генерується тривимірна координатна сітка X, Y, та Z. Потім задається тензорне поле у формі матриці розміром 3x3x3 з використанням np.array.

Функція plot\_tensor\_field(geometry, tensor\_field) приймає на вхід тип геометричної фігури geometry та тензорне поле tensor\_field. У циклі перебираються всі елементи тензорного поля, та в залежності від обраної геометрії візуалізуються відповідні геометричні фігури.

Викликається функція plot\_tensor\_field для кожної геометричної фігури: еліпсоїди, кубоїди, циліндри та суперквадри.

Зображення, що містить ряд, схема, знімок екрана, Графік

Автоматично згенерований опис

Зображення, що містить текст, схема, ряд, знімок екрана

Автоматично згенерований опис

Зображення, що містить схема, ряд, ескіз, дизайн

Автоматично згенерований опис

Зображення, що містить схема, ряд, Графік, текст

Автоматично згенерований опис

Виконала: Павленко Дарина

Група: ІКМ-М223в