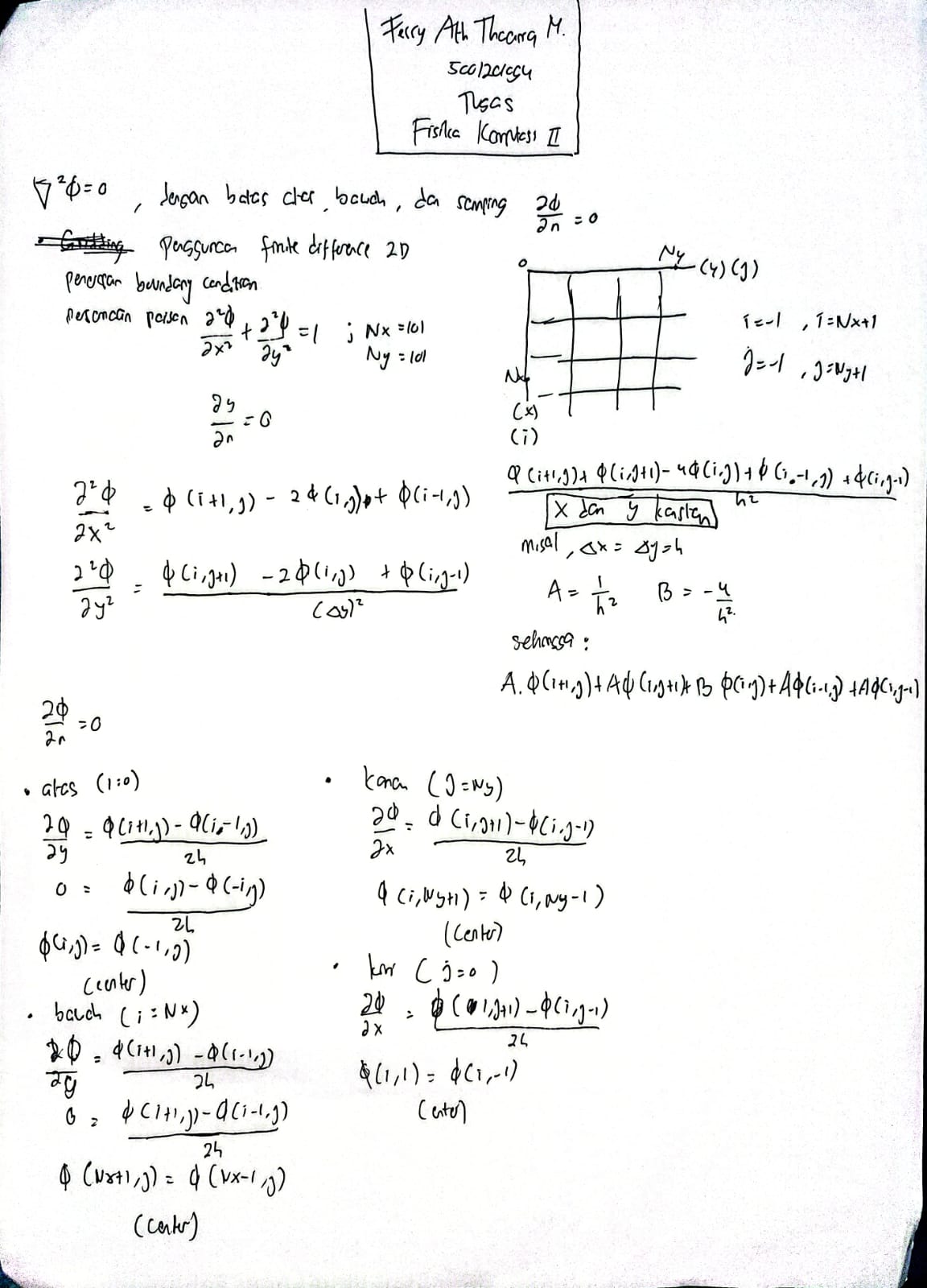
Ferry Ath Thaariq Mudhofir

Tugas

Fisika Komputasi II

∇2𝜙=0, dengan batas atas, bawah, dan samping = 0



**import** matplotlib.pyplot **as** plt  *# mengimport modul untuk visualisasi data*

**import** numpy **as** np  *# mengimport modul untuk manipulasi array dan matriks*

**import** math  *# mengimport modul matematika*

*# inisialisasikan data diketahui*

y **=** 1000  *# batas atas domain pada sumbu y*

x **=** 1000  *# batas atas domain pada sumbu x*

Ny **=** 101  *# jumlah node pada sumbu y*

Nx **=** 101  *# jumlah node pada sumbu x*

dy **=** y **/** (Ny**-**1)  *# jarak antara node pada sumbu y*

dx **=** x **/** (Nx**-**1)  *# jarak antara node pada sumbu x*

xa **=** np.arange(0, Nx**\***dx, dx)  *# array yang berisi koordinat x dari setiap node*

ya **=** np.arange(0, Ny**\***dy, dy)  *# array yang berisi koordinat y dari setiap node*

a **=** 1 **/** (dx**\*\***2)  *# konstanta pada persamaan Laplace*

c **=** 1 **/** (dy**\*\***2)  *# konstanta pada persamaan Laplace*

b **=** **-**2 **\*** (a **+** c)  *# konstanta pada persamaan Laplace*

*# definisikan matriks dengan nilai batas diketahui*

**def** create\_matrix(**a**, **b**, **c**, **Nx**, **Ny**):

    B **=** np.zeros((Nx**\***Ny, 1))  *# array nol untuk vektor batas*

    A **=** np.eye(Nx**\***Ny)  *# matriks identitas untuk matriks koefisien*

**for** i **in** range(1, Ny**-**1):

**for** j **in** range(1, Nx**-**1):

            k **=** (j**-**1)**\***Ny **+** i  *# indeks matriks dari node (i,j)*

            B[k, 0] **=** 1  *# nilai batas pada node (i,j)*

            A[k, k] **=** b  *# koefisien pada node (i,j)*

            A[k, k**-**1] **=** a  *# koefisien pada node sebelah kiri*

            A[k, k**+**1] **=** a  *# koefisien pada node sebelah kanan*

            A[k, k**-**Ny] **=** c  *# koefisien pada node di atas*

            A[k, k**+**Ny] **=** c  *# koefisien pada node di bawah*

**return** A, B

M, H **=** create\_matrix(a, b, c, Nx, Ny)  *# membuat matriks koefisien dan vektor batas*

sam **=** np.linalg.solve(M, H)  *# menyelesaikan sistem persamaan*

sol **=** np.zeros((Ny, Nx))  *# array kosong untuk solusi numerik*

**for** i **in** range(Ny):

**for** j **in** range(Nx):

        k **=** (j**-**1)**\***Ny **+** i  *# indeks matriks dari node (i,j)*

*# terjemahkan isi psi(x,y) pada matriks kolom [sam] ke [sol]*

        sol[i, j] **=** sam[k]

*# visualisasikan hasil*

X, Y **=** np.meshgrid(xa, ya)

plt.figure()

plt.pcolormesh(X, Y, sol, **shading=**'interp', **cmap=**'jet')

plt.colorbar()

plt.axis('equal')

plt.xlim([0, max(xa)])

plt.ylim([0, max(ya)])

plt.xlabel('X')

plt.ylabel('Y')

plt.show()

Kode Python di atas mengimplementasikan *finite element method* untuk menyelesaikan persamaan Laplace dengan nilai batas diketahui. Persamaan Laplace adalah persamaan diferensial parsial linear kedua yang menjelaskan distribusi lapangan potensial yang stabil di dalam suatu ruang tertentu.

Kode Python ini menggunakan library NumPy dan Matplotlib untuk menghasilkan visualisasi solusi persamaan Laplace pada suatu domain dua dimensi. NumPy digunakan untuk melakukan operasi matematika pada matriks, sedangkan Matplotlib digunakan untuk menghasilkan plot warna solusi.

Kode ini juga mengimplementasikan fungsi create\_matrix untuk membuat matriks koefisien sistem persamaan linear. Selain itu, kode ini juga menggunakan fungsi np.linalg.solve untuk menyelesaikan sistem persamaan linear dan menghasilkan solusi numerik.

Program Python ini menyelesaikan persamaan Laplace pada domain persegi dengan batas dirichlet homogen (yaitu nilai batas di setiap sisi domain adalah nol) menggunakan metode matriks.

Algoritma utama program ini adalah sebagai berikut:

1. Mengimport tiga modul yaitu matplotlib, numpy, dan math.
2. Menentukan batas atas pada sumbu y dan x (y dan x).
3. Menentukan jumlah node pada sumbu y dan x (Ny dan Nx).
4. Menghitung jarak antara setiap node pada sumbu y dan x (dy dan dx).
5. Membuat dua array xa dan ya yang berisi koordinat x dan y dari setiap node menggunakan fungsi arange dari modul numpy.
6. Menentukan konstanta a, b, dan c pada persamaan Laplace.
7. Membuat matriks koefisien (A) dan vektor batas (B) menggunakan fungsi create\_matrix yang mengambil konstanta a, b, dan c serta jumlah node pada sumbu y dan x sebagai argumen.
8. Menyelesaikan sistem persamaan linear menggunakan fungsi solve dari modul numpy.
9. Memasukkan nilai solusi numerik ke dalam array kosong sol menggunakan perulangan for.
10. Menampilkan visualisasi hasil menggunakan modul matplotlib dengan memplot solusi numerik sebagai peta warna (dengan menggunakan fungsi pcolormesh) dan menambahkan label sumbu x dan y serta skala warna.

