Assignment 3
Numerical solution of the two-dimensional Poisson's equation using the Finite-Difference and Finite-Volume Methods

Numerical Analysis For PDE's (WI3730TU)
Kaloyan Yanchev

September 2024

1. Boundary-Value Problem

   $-\nabla \cdot (k\nabla u) = f, (x, y) \in \Omega = (0, 10) \times (0, 5),$

   $u(x, y) = 0, (x, y) \in \partial\Omega,$

   $f(x, y) = \sum_{i=1}^{9} \sum_{j=1}^{4} e^{-\alpha(x-i)^2-\alpha(y-j)^2}, \alpha = 40, (x, y) \in \overline{\Omega}$

2. Finite-Difference Method

   $k(x, y) = 1, (x, y) \in \overline{\Omega}$

   2.1. Discretization

   2.1.1. $-\triangle u_{i,j} = \frac{-u_{i-1,j}+2u_{i,j}-u_{i+1,j}}{h_x^2} + \frac{-u_{i,j-1}+2u_{i,j}-u_{i,j+1}}{h_y^2} + O(h_x^2) + O(h_y^2)$

   2.1.2. Given $u(x, y) = 0, (x, y) \in \partial\Omega => u(x_i, y_j) = 0, (i, j) \in \{(0, 0), (0, 1), (0, 2), (0, 3), (0, 4), (1, 4), (2, 4), (3, 4), (4, 4), (4, 3), (4, 2), (4, 1), (4, 0), (3, 0), (2, 0), (1, 0)\}:$

$$\begin{cases} f_{1,1} = \frac{2u_{1,1}-u_{2,1}}{h_x^2} + \frac{2u_{1,1}-u_{1,2}}{h_y^2} \\ f_{2,1} = \frac{-u_{1,1}+2u_{2,1}-u_{3,1}}{h_x^2} + \frac{2u_{2,1}-u_{2,2}}{h_y^2} \\ f_{3,1} = \frac{-u_{2,1}+2u_{3,1}}{h_x^2} + \frac{2u_{3,1}-u_{3,2}}{h_y^2} \\ f_{1,2} = \frac{2u_{1,2}-u_{2,2}}{h_x^2} + \frac{-u_{1,1}+2u_{1,2}-u_{1,3}}{h_y^2} \\ f_{2,2} = \frac{-u_{1,2}+2u_{2,2}-u_{3,2}}{h_x^2} + \frac{-u_{2,1}+2u_{2,2}-u_{2,3}}{h_y^2} \\ f_{3,2} = \frac{-u_{2,2}+2u_{3,2}}{h_x^2} + \frac{-u_{3,1}+2u_{3,2}-u_{3,3}}{h_y^2} \\ f_{1,3} = \frac{2u_{1,3}-u_{2,3}}{h_x^2} + \frac{-u_{1,2}+2u_{1,3}}{h_y^2} \\ f_{2,3} = \frac{-u_{1,3}+2u_{2,3}-u_{3,3}}{h_x^2} + \frac{-u_{2,2}+2u_{2,3}}{h_y^2} \\ f_{3,3} = \frac{-u_{2,3}+2u_{3,3}}{h_x^2} + \frac{-u_{3,2}+2u_{3,3}}{h_y^2} \end{cases}$$

   2.1.3. $A\mathbf{u} = \mathbf{f}$

$$A = \begin{bmatrix} \frac{2}{h_x^2}+\frac{2}{h_y^2} & -\frac{1}{h_x^2} & 0 & -\frac{1}{h_y^2} & 0 & 0 & 0 & 0 & 0 \\ -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2} & -\frac{1}{h_x^2} & 0 & -\frac{1}{h_y^2} & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2} & -\frac{1}{h_x^2} & 0 & -\frac{1}{h_y^2} & 0 & 0 & 0 \\ -\frac{1}{h_y^2} & 0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2} & -\frac{1}{h_x^2} & 0 & -\frac{1}{h_y^2} & 0 & 0 \\ 0 & -\frac{1}{h_y^2} & 0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2} & -\frac{1}{h_x^2} & 0 & -\frac{1}{h_y^2} & 0 \\ 0 & 0 & -\frac{1}{h_y^2} & 0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2} & -\frac{1}{h_x^2} & 0 & -\frac{1}{h_y^2} \\ 0 & 0 & 0 & -\frac{1}{h_y^2} & 0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2} & -\frac{1}{h_x^2} & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{h_y^2} & 0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2} & -\frac{1}{h_x^2} \\ 0 & 0 & 0 & 0 & 0 & -\frac{1}{h_y^2} & 0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2} \end{bmatrix}$$

$h_x = \frac{10-0}{4} = 2.5 => \frac{1}{h_x} = 0.4 => \frac{1}{h_x^2} = 0.16$

$h_y = \frac{5-0}{4} = 1.25 => \frac{1}{h_y} = 0.8 => \frac{1}{h_y^2} = 0.64$

$$A = \begin{bmatrix} 1.6 & -0.16 & 0 & -0.64 & 0 & 0 & 0 & 0 & 0 \\ -0.16 & 1.6 & -0.16 & 0 & -0.64 & 0 & 0 & 0 & 0 \\ 0 & -0.16 & 1.6 & -0.16 & 0 & -0.64 & 0 & 0 & 0 \\ -0.64 & 0 & -0.16 & 1.6 & -0.16 & 0 & -0.64 & 0 & 0 \\ 0 & -0.64 & 0 & -0.16 & 1.6 & -0.16 & 0 & -0.64 & 0 \\ 0 & 0 & -0.64 & 0 & -0.16 & 1.6 & -0.16 & 0 & -0.64 \\ 0 & 0 & 0 & -0.64 & 0 & -0.16 & 1.6 & -0.16 & 0 \\ 0 & 0 & 0 & 0 & -0.64 & 0 & -0.16 & 1.6 & -0.16 \\ 0 & 0 & 0 & 0 & 0 & -0.64 & 0 & -0.16 & 1.6 \end{bmatrix}$$

$$\mathbf{u} = \begin{bmatrix} u_{1,1} \\ u_{2,1} \\ u_{3,1} \\ u_{1,2} \\ u_{2,2} \\ u_{3,2} \\ u_{1,3} \\ u_{2,3} \\ u_{3,3} \end{bmatrix}, \quad \mathbf{f} = \begin{bmatrix} f_{1,1} \\ f_{2,1} \\ f_{3,1} \\ f_{1,2} \\ f_{2,2} \\ f_{3,2} \\ f_{1,3} \\ f_{2,3} \\ f_{3,3} \end{bmatrix}$$

2.1.4. $D_x = \frac{1}{h_x}\begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \end{bmatrix}, \quad D_x^T = \frac{1}{h_x}\begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}, \quad L_{xx} = D_x^T D_x = \frac{1}{h_x^2}\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$

$D_y = \frac{1}{h_y}\begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \end{bmatrix}, \quad D_y^T = \frac{1}{h_y}\begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}, \quad L_{yy} = D_y^T D_y = \frac{1}{h_y^2}\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$

$A = I_y \otimes L_{xx} + L_{yy} \otimes I_x$

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} \frac{2}{h_x^2} & -\frac{1}{h_x^2} & 0 \\ -\frac{1}{h_x^2} & \frac{2}{h_x^2} & -\frac{1}{h_x^2} \\ 0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2} \end{bmatrix} + \begin{bmatrix} \frac{2}{h_y^2} & -\frac{1}{h_y^2} & 0 \\ -\frac{1}{h_y^2} & \frac{2}{h_y^2} & -\frac{1}{h_y^2} \\ 0 & -\frac{1}{h_y^2} & \frac{2}{h_y^2} \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$A = \begin{bmatrix} \frac{2}{h_x^2} & -\frac{1}{h_x^2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{1}{h_x^2} & \frac{2}{h_x^2} & -\frac{1}{h_x^2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2} & -\frac{1}{h_x^2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2} & -\frac{1}{h_x^2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2} & -\frac{1}{h_x^2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2} & -\frac{1}{h_x^2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2} & -\frac{1}{h_x^2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2} & -\frac{1}{h_x^2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2} \end{bmatrix} +$$

$$\begin{bmatrix} \frac{2}{h_y^2} & 0 & 0 & -\frac{1}{h_y^2} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{2}{h_y^2} & 0 & 0 & -\frac{1}{h_y^2} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{2}{h_y^2} & 0 & 0 & -\frac{1}{h_y^2} & 0 & 0 & 0 \\ -\frac{1}{h_y^2} & 0 & 0 & \frac{2}{h_y^2} & 0 & 0 & -\frac{1}{h_y^2} & 0 & 0 \\ 0 & -\frac{1}{h_y^2} & 0 & 0 & \frac{2}{h_y^2} & 0 & 0 & -\frac{1}{h_y^2} & 0 \\ 0 & 0 & -\frac{1}{h_y^2} & 0 & 0 & \frac{2}{h_y^2} & 0 & 0 & -\frac{1}{h_y^2} \\ 0 & 0 & 0 & -\frac{1}{h_y^2} & 0 & 0 & \frac{2}{h_y^2} & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{h_y^2} & 0 & 0 & \frac{2}{h_y^2} & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{1}{h_y^2} & 0 & 0 & \frac{2}{h_y^2} \end{bmatrix}$$

$$A = \begin{bmatrix} \frac{2}{h_x^2}+\frac{2}{h_y^2} & -\frac{1}{h_x^2} & 0 & -\frac{1}{h_y^2} & 0 & 0 & 0 & 0 & 0 \\ -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2} & -\frac{1}{h_x^2} & 0 & -\frac{1}{h_y^2} & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2} & -\frac{1}{h_x^2} & 0 & -\frac{1}{h_y^2} & 0 & 0 & 0 \\ -\frac{1}{h_y^2} & 0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2} & -\frac{1}{h_x^2} & 0 & -\frac{1}{h_y^2} & 0 & 0 \\ 0 & -\frac{1}{h_y^2} & 0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2} & -\frac{1}{h_x^2} & 0 & -\frac{1}{h_y^2} & 0 \\ 0 & 0 & -\frac{1}{h_y^2} & 0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2} & -\frac{1}{h_x^2} & 0 & -\frac{1}{h_y^2} \\ 0 & 0 & 0 & -\frac{1}{h_y^2} & 0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2} & -\frac{1}{h_x^2} & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{h_y^2} & 0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2} & -\frac{1}{h_x^2} \\ 0 & 0 & 0 & 0 & 0 & -\frac{1}{h_y^2} & 0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2} \end{bmatrix}$$

2.2. Implementation

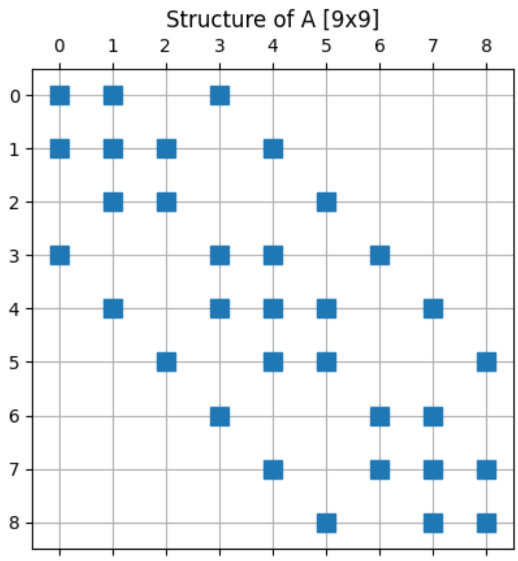
2.2.1. A.

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.sparse as sp
import scipy.sparse.linalg as la
```

B.

```
LeftX = 0.0
RightX = 10.0
LeftY = 0.0
RightY = 5.0

Nx = 4  # number of intervals in x-direction
Ny = 4  # number of intervals in y-direction
dx = (RightX - LeftX) / Nx  # grid step in x-direction
dy = (RightY - LeftY) / Ny  # grid step in y-direction
```

C. The values do indeed match except for the fact that the 16-th digit after the decimal is already experiencing a floating point rounding error.



D.
```python
def FDLaplacian2D(nx: int, ny: int):

    hx = (RightX - LeftX) / nx  # grid step in x-direction
    hy = (RightY - LeftY) / ny  # grid step in y-direction

    Dx = sp.diags([-1, 1], (-1, 0), (nx, nx-1)) / hx
    Dy = sp.diags([-1, 1], (-1, 0), (ny, ny-1)) / hy

    DxT = Dx.transpose()
    DyT = Dy.transpose()

    Lxx = DxT.dot(Dx)
    Lyy = DyT.dot(Dy)

    Ix = sp.eye(nx-1)
    Iy = sp.eye(ny-1)

    A = sp.kron(Iy, Lxx) + sp.kron(Lyy, Ix)

    return A
```

2.2.2. A.
```python
def sourcefunc(x, y):
    f = 0.0
    alpha = 40.0
    x = np.array(x)
    y = np.array(y)
    for i in range(1, 10):
        for j in range(1, 5):
            f += np.exp(-alpha*(x-i)**2-alpha*(y-j)**2)
    return f
```

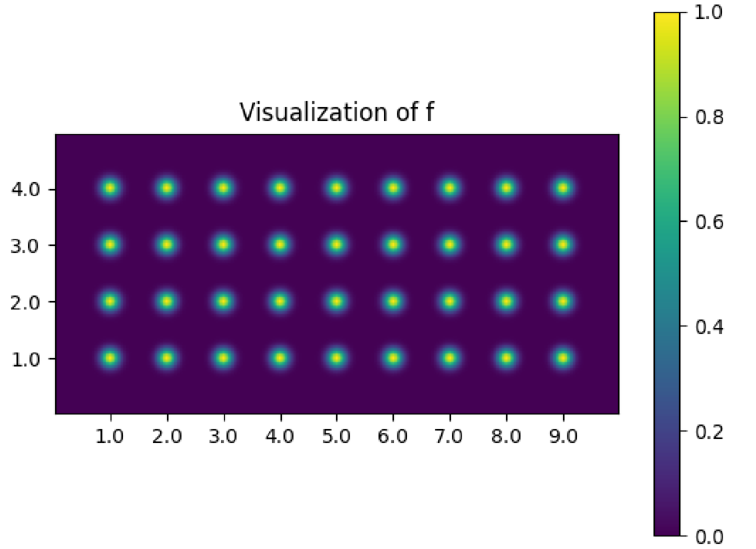
B.
```python
x, y = np.mgrid[1:Nx, 1:Ny]
x = x.astype('float64')
x *= dx
y = y.astype('float64')
y *= dy
```

The structure of x and y represents the lexicographical order in the sense that x[0] contains only x_1 values, while y[0] contains all values from y_1 to y_{N_y-1}, however, the lexicographical ordering takes elements $a_{1,1}, a_{2,1}, ..., a_{N_x-1,1}, ...$, therefore the values obtained using these matrices need to be inverted. Thus the first array dimension represents the y-direction (x-array contains the same value, while y-array contains all y-values) and the second one - the x-direction (x-array contains all x-values, while y-array contains the same value).

C.
```python
f = sourcefunc(x, y).transpose()
```

2.2.3. A.
```python
def vis_arr(func: np.ndarray, x_val: np.ndarray,
            y_val: np.ndarray, title: str):
    plt.ion()
    plt.figure(1)
    plt.clf()
    plt.imshow(func)
    plt.title(title)
    plt.colorbar()

    # Invert y-axis
    ax = plt.gca()
    ax.set_ylim(ax.get_ylim()[::-1])

    # Scale x,y ranges
    plt.xticks(range(19, Nx-1, 20), np.round(x_val[19::20, 0], 2))
```

```
    plt.yticks(range(19, Ny-1, 20), np.round(y_val[0, 19::20], 2))
    plt.show()

vis_arr(f, x, y, "Visualization of f")
```



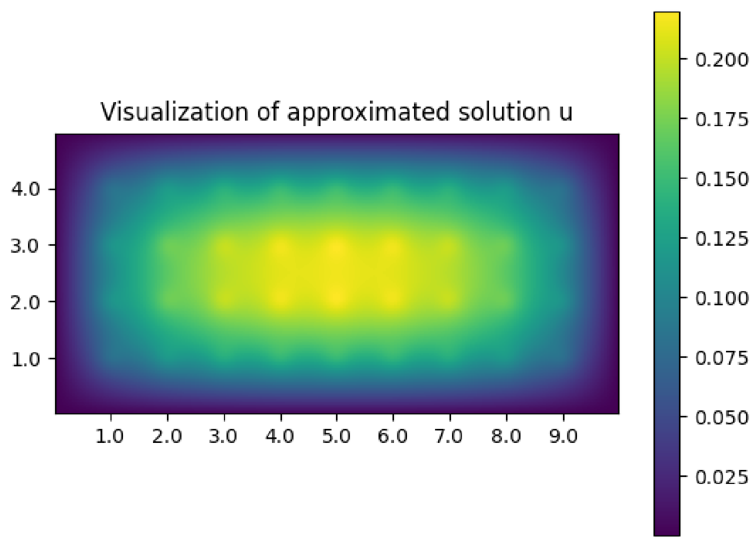
B.
```
# lexicographic source vector
fLX = np.reshape(f, ((Nx-1) * (Ny-1)))
```

C.
```
# 2D FD Laplacian on rectangular domain
A = FDLaplacian2D(200, 100)
```

D. The vector **u** contains the approximations of u at the inner grid points in lexicographic order.

```
u = la.spsolve(A, fLX)
```

E.
```
# reshaping the solution vector into 2D array
uArr = np.reshape(u, (Ny-1, Nx-1))

vis_arr(uArr, x, y, "Visualization of approximated solution u")
```



3. Finite-Volume Method

$k(x, y) = 1 + 0.1(x + y + xy), (x, y) \in \overline{\Omega}$

3.1. Discretization

3.1.1. Given $u(x, y) = 0, (x, y) \in \partial\Omega \Rightarrow u(x_i, y_j) = 0, (i, j) \in \{(0, 0), (0, 1), (0, 2), (0, 3), (0, 4), (1, 4), (2, 4), (3, 4), (4, 4), (4, 3), (4, 2), (4, 1), (4, 0), (3, 0), (2, 0), (1, 0)\}$:

$$\begin{cases}
f_{1,1} = (\frac{k_{0.5,1}}{h_x^2} + \frac{k_{1,0.5}}{h_y^2} + \frac{k_{1.5,1}}{h_x^2} + \frac{k_{1,1.5}}{h_y^2})u_{1,1} - \frac{k_{1.5,1}}{h_x^2}u_{2,1} - \frac{k_{1,1.5}}{h_y^2}u_{1,2} \\
f_{2,1} = -\frac{k_{1.5,1}}{h_x^2}u_{1,1} + (\frac{k_{1.5,1}}{h_x^2} + \frac{k_{2,0.5}}{h_y^2} + \frac{k_{2.5,1}}{h_x^2} + \frac{k_{2,1.5}}{h_y^2})u_{2,1} - \frac{k_{2.5,1}}{h_x^2}u_{3,1} - \frac{k_{2,1.5}}{h_y^2}u_{2,2} \\
f_{3,1} = -\frac{k_{2.5,1}}{h_x^2}u_{2,1} + (\frac{k_{2.5,1}}{h_x^2} + \frac{k_{3,0.5}}{h_y^2} + \frac{k_{3.5,1}}{h_x^2} + \frac{k_{3,1.5}}{h_y^2})u_{3,1} - \frac{k_{3,1.5}}{h_y^2}u_{3,2} \\
f_{1,2} = -\frac{k_{1,1.5}}{h_y^2}u_{1,1} + (\frac{k_{0.5,2}}{h_x^2} + \frac{k_{1,1.5}}{h_y^2} + \frac{k_{1.5,2}}{h_x^2} + \frac{k_{1,2.5}}{h_y^2})u_{1,2} - \frac{k_{1.5,2}}{h_x^2}u_{2,2} - \frac{k_{1,2.5}}{h_y^2}u_{1,3} \\
f_{2,2} = -\frac{k_{1.5,2}}{h_x^2}u_{1,2} - \frac{k_{2,1.5}}{h_y^2}u_{2,1} + (\frac{k_{1.5,2}}{h_x^2} + \frac{k_{2,1.5}}{h_y^2} + \frac{k_{2.5,2}}{h_x^2} + \frac{k_{2,2.5}}{h_y^2})u_{2,2} - \frac{k_{2.5,2}}{h_x^2}u_{3,2} - \frac{k_{2,2.5}}{h_y^2}u_{2,3} \\
f_{3,2} = -\frac{k_{2.5,2}}{h_x^2}u_{2,2} - \frac{k_{3,1.5}}{h_y^2}u_{3,1} + (\frac{k_{2.5,2}}{h_x^2} + \frac{k_{2,1.5}}{h_y^2} + \frac{k_{3.5,2}}{h_x^2} + \frac{k_{3,2.5}}{h_y^2})u_{3,2} - \frac{k_{3,2.5}}{h_y^2}u_{3,3} \\
f_{1,3} = -\frac{k_{1,2.5}}{h_y^2}u_{1,2} + (\frac{k_{0.5,3}}{h_x^2} + \frac{k_{1,2.5}}{h_y^2} + \frac{k_{1.5,3}}{h_x^2} + \frac{k_{1,3.5}}{h_y^2})u_{1,3} - \frac{k_{1.5,3}}{h_x^2}u_{2,3} \\
f_{2,3} = -\frac{k_{1.5,3}}{h_x^2}u_{1,3} - \frac{k_{2,2.5}}{h_y^2}u_{2,2} + (\frac{k_{1.5,3}}{h_x^2} + \frac{k_{2,2.5}}{h_y^2} + \frac{k_{2.5,3}}{h_x^2} + \frac{k_{2,3.5}}{h_y^2})u_{2,3} - \frac{k_{2.5,3}}{h_x^2}u_{3,3} \\
f_{3,3} = -\frac{k_{2.5,3}}{h_x^2}u_{2,3} - \frac{k_{3,2.5}}{h_y^2}u_{3,2} + (\frac{k_{2.5,3}}{h_x^2} + \frac{k_{2,2.5}}{h_y^2} + \frac{k_{3.5,3}}{h_x^2} + \frac{k_{3,3.5}}{h_y^2})u_{3,3}
\end{cases}$$

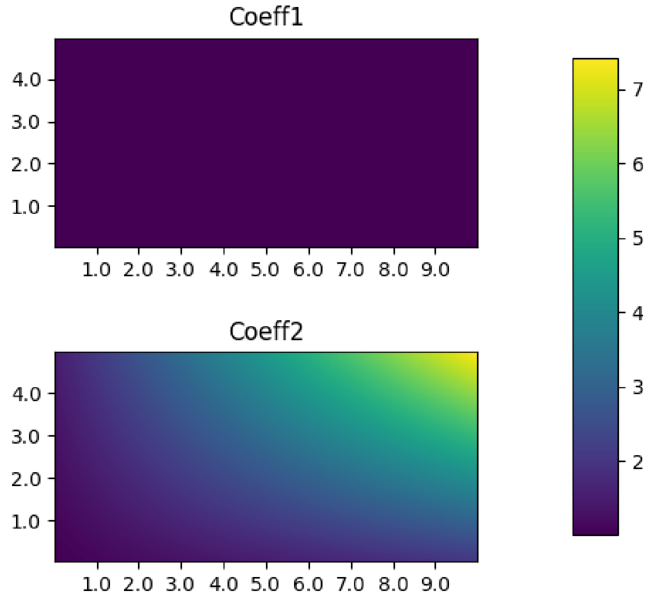
3.1.2. Given $k(x, y) = 1, (x, y) \in \overline{\Omega}$:

$$A = \begin{bmatrix}
\frac{2}{h_x^2}+\frac{2}{h_y^2} & -\frac{1}{h_x^2} & 0 & -\frac{1}{h_y^2} & 0 & 0 & 0 & 0 & 0 \\
-\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2} & -\frac{1}{h_x^2} & 0 & -\frac{1}{h_y^2} & 0 & 0 & 0 & 0 \\
0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2} & -\frac{1}{h_x^2} & 0 & -\frac{1}{h_y^2} & 0 & 0 & 0 \\
-\frac{1}{h_y^2} & 0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2} & -\frac{1}{h_x^2} & 0 & -\frac{1}{h_y^2} & 0 & 0 \\
0 & -\frac{1}{h_y^2} & 0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2} & -\frac{1}{h_x^2} & 0 & -\frac{1}{h_y^2} & 0 \\
0 & 0 & -\frac{1}{h_y^2} & 0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2} & -\frac{1}{h_x^2} & 0 & -\frac{1}{h_y^2} \\
0 & 0 & 0 & -\frac{1}{h_y^2} & 0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2} & -\frac{1}{h_x^2} & 0 \\
0 & 0 & 0 & 0 & -\frac{1}{h_y^2} & 0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2} & -\frac{1}{h_x^2} \\
0 & 0 & 0 & 0 & 0 & -\frac{1}{h_y^2} & 0 & -\frac{1}{h_x^2} & \frac{2}{h_x^2}+\frac{2}{h_y^2}
\end{bmatrix}$$

This is indeed the same matrix as derived in 2.1.3.

3.2. Implementation

3.2.1. A.
```
def coeffK1(x, y):
    x = np.array(x)
    K = np.ones_like(x)
    return K

def coeffK2(x, y):
    x = np.array(x)
    y = np.array(y)
    K = 1 + 0.1 * (x + y + x*y)
    return K
```



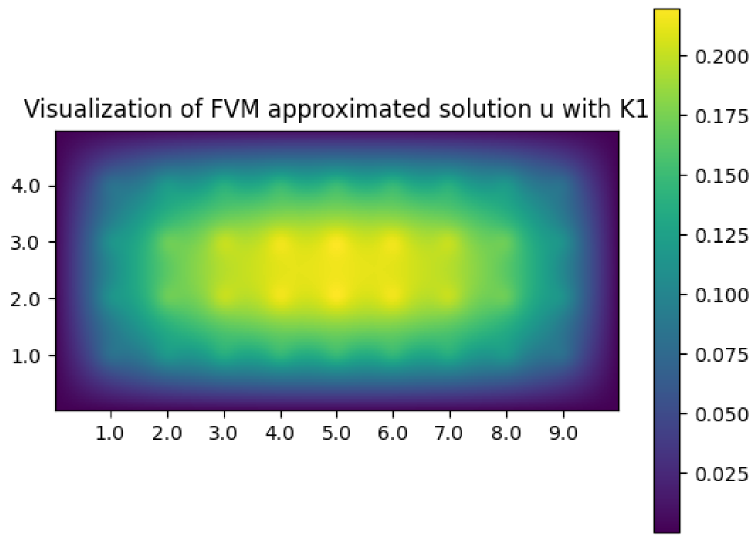
B.
```
def create2DLFVM(nx: int, ny: int, coeffFun, out: bool = False):
    hx = (RightX - LeftX) / nx  # grid step in x-direction
    hy = (RightY - LeftY) / ny  # grid step in y-direction

    kx = (RightX - LeftX) / (2*nx)
    ky = (RightY - LeftY) / (2*ny)

    x, y = np.mgrid[1:2*nx, 1:2*ny]
    x = x.astype('float64')
    x *= kx
    y = y.astype('float64')
    y *= ky

    k = coeffFun(x, y)
    main_d = []
    x_d = []
    y_d = []

    for j in range(1, ny):
        for i in range(1, nx):
            main_d.append(k[2*i-2, 2*j-1]/hx**2
                          + k[2*i-1, 2*j-2]/hy**2
                          + k[2*i, 2*j-1]/hx**2
                          + k[2*i-1, 2*j]/hy**2)
            if i < nx-1:
                x_d.append(-k[2*i, 2*j-1]/hx**2)
            else:
                x_d.append(0)
            y_d.append(-k[2*i-1, 2*j]/hy**2)

    A = sp.diags([y_d, x_d, main_d, x_d, y_d],
                 [-2*ny+1, -1, 0, 1, 2*ny-1],
                 ((nx-1)*(ny-1), (nx-1)*(ny-1)), format='csc')

    if out:
        print(A)

    return A
```

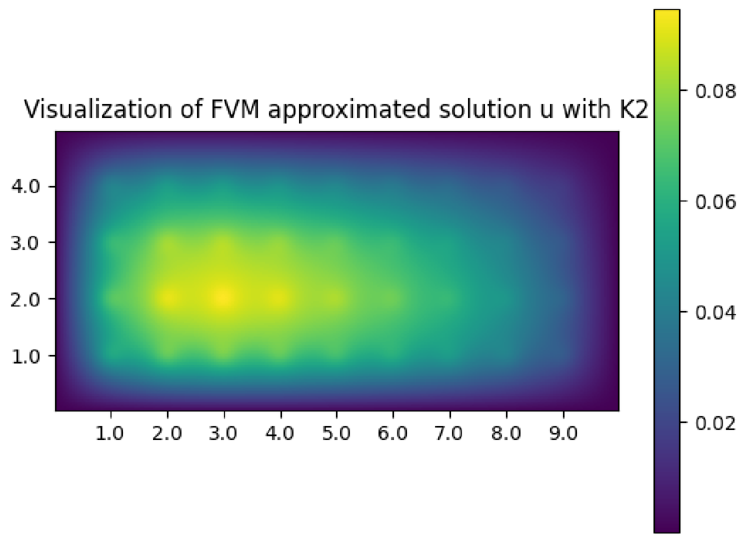
C. The matrix A derived with $N_x = 4,\ N_y = 4,\ k = 1, (x, y) \in \overline{\Omega}$ using the FVM matrix computation does indeed produce the same matrix as the FDM one computed in 2.1.3.

D. Same variable is used as the values of $\mathbf{f}$ are the same.

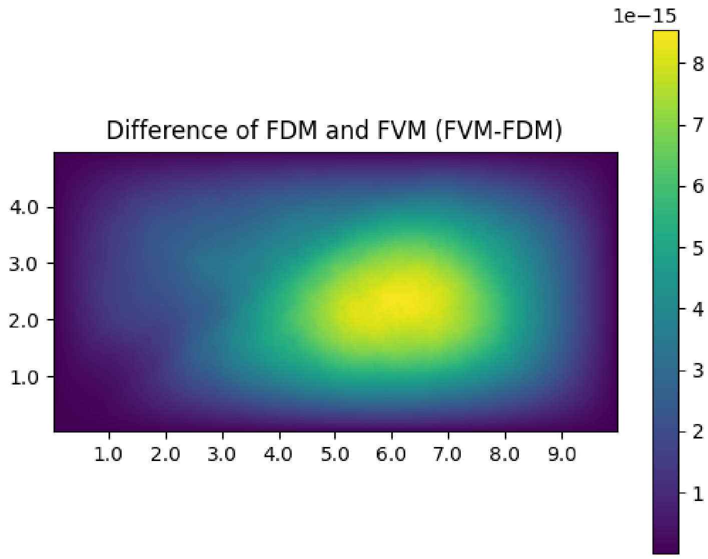
3.2.2. A.

```
A_K1 = create2DLFVM(Nx, Ny, coeffK1)
A_K2 = create2DLFVM(Nx, Ny, coeffK2)

u_K1 = la.spsolve(A_K1, fLX)
u_K2 = la.spsolve(A_K2, fLX)

uArr_K1 = np.reshape(u_K1, (Ny - 1, Nx - 1))
uArr_K2 = np.reshape(u_K2, (Ny - 1, Nx - 1))

vis_arr(uArr_K1, x, y,
        "Visualization of FVM approximated solution u with K1")
vis_arr(uArr_K2, x, y,
        "Visualization of FVM approximated solution u with K2")
```





4. Given that f experiences values approaching 1 around integer coordinated points (for example (3,4)) and for the most part of the rest of the domain is approaching 0, it is expected, as can be observed in the results, that the function u also experiences local maximums at these points. Similarly, given that f is non-negative and the boundary condition of u is set to 0, it follows that u contains only non-negative values. For constant k, it can also be expected, as is observed, that the further away from the boundary a point is, the higher its value is. The difference between the results of FDM and FVM can be computed and visualized.



As it can be seen, the difference is of magnitude 10^{-15} which is most likely due to floating point error, especially given that FVM performs more computations per point that FDM. Concerning non-uniform k, it can be expected, given that there is a uniform boundary condition and k and u have an inverse relationship, that the region with higher k values, would have lower u values.
Physically, the problem can be interpreted by saying u represents the temperature of the surface, f is a heat source, and k is the thermal conductivity of the material, with boundary at constant 0 degrees meaning the system is emitting heat into the environment. If the thermal conductivity is constant, it is expected that points around the heat sources will be hotter than their surroundings, and the region that is furthest away from any boundaries would conserve more heat than others closer to the boundary. Also given that there is a positive heat source on the surface and heat is only flowing out of the system, it should be expected that the system experiences only non-negative temperatures. In the cases, when k is non-uniform, meaning that there are more conductive regions than others, it would be expected those more conductive regions around the boundary emit more heat out of the system, thus being less hot than other less conductive regions that would conserve more heat in the system.