Shraddha Raghuram
5211972
ShraddhaRaghuram@student.tudelft.nl

Assignment 3:
Numerical Solution of the Two-Dimensional Poisson's Equation Using the Finite-Difference and Finite-Volume Methods

2022-09-30

1 Boundary-Value Problem

Consider the following boundary value problem:

$$-\nabla \cdot (k\nabla u) = f, (x,y) \in \Omega = (0,10) \times (0,5),$$
$$u(x,y) = 0, (x,y) \in \partial\Omega,$$
$$f(x,y) = \sum_{i=1}^{9}\sum_{j=1}^{4} e^{-\alpha(x-i)^2-\alpha(y-j)^2} \tag{1}$$

with $\alpha = 40, (x,y) \in \overline{\Omega}$, where $\overline{\Omega} = [0,10] \times [0,5]$ is the rectangle with the corners (0,0), (10,0), (10,5), and (0,5).

2 Finite-Difference Method

For this method we shall only consider the homogeneous coefficient function

$$k(x,y) = 1, (x,y) \in \overline{\Omega} \tag{2}$$

2.1 Discretization

First, we explicitly discretize the problem on a small mesh with 4 steps (sub- intervals) in both the x- and the y-directions and adopt the lexicographic ordering of the nodes and unknowns.

1. Write down the $\mathcal{O}(h^2)$ Finite-Difference (FD) approximation of $-\Delta u_{i,j}$ at a point (x_i, y_j) for a uniform, but not doubly-uniform mesh.

   $$-\nabla \cdot (k\nabla u) = -\nabla k \nabla u - k\delta u = f$$

   2D grid on $\Omega = (0,10) \times (0,5)$ is built from two uniform 1D grids

   $$\Omega_h = (x_i, y_j) | x_i = 0 + ih_x, i = 0,1,....,4; y_j = 0 + jh_y, j = 0,1,....,4$$

   where

   $$h_x = \frac{b-a}{N_x} = \frac{10-0}{4} = \frac{10}{4}$$
   $$h_y = \frac{d-c}{N_y} = \frac{5-0}{4} = \frac{5}{4}$$

   Grid values are $u_{i,j} = u(x_i, y_j)$ and $f_{i,j} = f(x_i, y_j)$. The Finite-Difference approximation of a negative 2D Laplacian is as follows:

   $$-\frac{\partial^2 u_{i,j}}{\partial x^2} = \frac{-u_{i-1,j} + 2u_{i,j} - u_{i+1,j}}{h_x^2} + \mathcal{O}(h_x^2) == \frac{-u_{i-1,j} + 2u_{i,j} - u_{i+1,j}}{\frac{100}{16}} + \mathcal{O}(h_x^2)$$

   $$-\frac{\partial^2 u_{i,j}}{\partial y^2} = \frac{-u_{i,j-1} + 2u_{i,j} - u_{i,j+1}}{h_y^2} + \mathcal{O}(h_y^2) = \frac{-u_{i,j-1} + 2u_{i,j} - u_{i,j+1}}{\frac{25}{16}} + \mathcal{O}(h_y^2)$$

   $$-\Delta u_{i,j} = -\frac{\partial^2 u_{i,j}}{\partial x^2} - \frac{\partial^2 u_{i,j}}{\partial y^2} = \frac{-u_{i-1,j} + 2u_{i,j} - u_{i+1,j}}{\frac{100}{16}} + \mathcal{O}(h_x^2) + \frac{-u_{i,j-1} + 2u_{i,j} - u_{i,j+1}}{\frac{25}{16}} + \mathcal{O}(h_y^2)$$

2. For your mesh (see above), write down all the discrete FD equations of your problem, performing the substitution and elimination of the boundary points and values. The node values of the source function f (x, y) may be written symbolically as $f_{i,j}$, without substituting the actual algebraic expression from (1).
Inner points:

$$\frac{-u_{i-1,j}+2u_{i,j}-u_{i+1,j}}{\frac{100}{16}}+\frac{-u_{i,j-1}+2u_{i,j}-u_{i,j+1}}{\frac{25}{16}}=f_{i,j}$$

i=1,2,....,3, j=1,2,....,3
Boundary points (homogeneous Dirichlet BC's u(x,y)=0, (x,y) $\in \partial\Omega$):

$$u_{i,j}=0$$

, if lower boundary (south): i=0,1,....,4;j=0
upper boundary (north): i=0,1,....,4;j=4
left boundary (west): i=0; j=0,1,....,4
right boundary (east): i=4; j=0,1,....,4
for i=1, j=1:

$$\frac{-u_{0,1}+2u_{1,1}-u_{2,1}}{\frac{100}{16}}+\frac{-u_{1,0}+2u_{1,1}-u_{1,2}}{\frac{25}{16}}=f_{1,1}$$

for i=1, j=2:

$$\frac{-u_{0,2}+2u_{1,2}-u_{2,2}}{\frac{100}{16}}+\frac{-u_{1,1}+2u_{1,2}-u_{1,3}}{\frac{25}{16}}=f_{1,2}$$

for i=1, j=3:

$$\frac{-u_{0,3}+2u_{1,3}-u_{2,3}}{\frac{100}{16}}+\frac{-u_{1,2}+2u_{1,3}-u_{1,4}}{\frac{25}{16}}=f_{1,3}$$

for i=2, j=1:

$$\frac{-u_{1,1}+2u_{2,1}-u_{3,1}}{\frac{100}{16}}+\frac{-u_{2,0}+2u_{2,1}-u_{2,2}}{\frac{25}{16}}=f_{2,1}$$

for i=2, j=2:

$$\frac{-u_{1,2}+2u_{2,2}-u_{3,2}}{\frac{100}{16}}+\frac{-u_{2,1}+2u_{2,2}-u_{2,3}}{\frac{25}{16}}=f_{2,2}$$

for i=2, j=3:

$$\frac{-u_{1,3}+2u_{2,3}-u_{3,3}}{\frac{100}{16}}+\frac{-u_{2,2}+2u_{2,3}-u_{2,4}}{\frac{25}{16}}=f_{2,3}$$

for i=3, j=1:

$$\frac{-u_{2,1}+2u_{3,1}-u_{4,1}}{\frac{100}{16}}+\frac{-u_{3,0}+2u_{3,1}-u_{3,2}}{\frac{25}{16}}=f_{3,1}$$

for i=3, j=2:

$$\frac{-u_{2,2}+2u_{3,2}-u_{4,2}}{\frac{100}{16}}+\frac{-u_{3,1}+2u_{3,2}-u_{3,3}}{\frac{25}{16}}=f_{3,2}$$

for i=3, j=3:

$$\frac{-u_{2,3}+2u_{3,3}-u_{4,3}}{\frac{100}{16}}+\frac{-u_{3,2}+2u_{3,3}-u_{3,4}}{\frac{25}{16}}=f_{3,3}$$

3. The FD equations can be written as the linear algebraic problem of the form:

$$A\mathbf{u}=f \tag{3}$$

Write down the system matrix A of the negative FD Laplacian, the vector $\mathbf{u}$ of unknowns and the right-hand-side (RHS) vector $\mathbf{f}$.
System matrix,

$$A=\begin{bmatrix}
\frac{8}{5} & \frac{-16}{100} & 0 & \frac{-16}{25} & 0 & 0 & 0 & 0 & 0 \\
\frac{-16}{100} & \frac{8}{5} & \frac{-16}{100} & 0 & \frac{-16}{25} & 0 & 0 & 0 & 0 \\
0 & \frac{-16}{100} & \frac{8}{5} & 0 & 0 & \frac{-16}{25} & 0 & 0 & 0 \\
\frac{-16}{25} & 0 & 0 & \frac{8}{5} & \frac{-16}{100} & 0 & \frac{-16}{25} & 0 & 0 \\
0 & \frac{-16}{25} & 0 & \frac{-16}{100} & \frac{8}{5} & \frac{-16}{100} & 0 & \frac{-16}{25} & 0 \\
0 & 0 & \frac{-16}{25} & 0 & \frac{-16}{100} & \frac{8}{5} & 0 & 0 & \frac{-16}{25} \\
0 & 0 & 0 & \frac{-16}{25} & 0 & 0 & \frac{8}{5} & \frac{-16}{100} & 0 \\
0 & 0 & 0 & 0 & \frac{-16}{25} & 0 & \frac{-16}{100} & \frac{8}{5} & \frac{-16}{100} \\
0 & 0 & 0 & 0 & 0 & \frac{-16}{25} & 0 & \frac{-16}{100} & \frac{8}{5}
\end{bmatrix}$$

Vector of the unknowns,

$$u = \begin{bmatrix} u_{1,1} \\ u_{2,1} \\ u_{3,1} \\ u_{1,2} \\ u_{2,2} \\ u_{3,2} \\ u_{1,3} \\ u_{2,3} \\ u_{3,3} \end{bmatrix}$$

Right hand side vector,

$$f = \begin{bmatrix} f_{1,1} \\ f_{2,1} \\ f_{3,1} \\ f_{1,2} \\ f_{2,2} \\ f_{3,2} \\ f_{1,3} \\ f_{2,3} \\ f_{3,3} \end{bmatrix}$$

4. Show for your grid that, under lexicographic ordering, the FD matrix A of the negative 2D Laplacian operator can indeed be obtained as A = $I_y \bigotimes L_{xx} + L_{yy} \bigotimes I_x$, where L_{xx} and L_{yy} are the FD matrices of negative 1D Laplacians in the x- and y-direction, respectively, and I_x, I_y are identity matrices of proper size. Also show that $L_{xx} = D_x^T D_x$ and $L_{yy} = D_y^T D_y$, where D_x and D_y are matrices representing one-sided FD approximations of the first derivatives. See slides of Lecture 5. You need to demonstrate that the Kronecker-product formula works.
Let $D_x \in \mathbb{R}^{4\times 3}$ and $D_y \in \mathbb{R}^{4\times 3}$ be the sparse first-order (partial) derivative matrices.

$$D_x = \frac{1}{h_x}\begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \end{bmatrix} = \frac{4}{10}\begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \end{bmatrix}$$

$$D_y = \frac{1}{h_y}\begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \end{bmatrix} = \frac{4}{5}\begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \end{bmatrix}$$

Define two sparse (partial) 1D Laplacian matrices as $L_{xx} = D_x^T D_x$ and $L_{yy} = D_y^T D_y$ with $L_{xx} \in \mathbb{R}^{3\times 3}$ and $L_{yy} \in \mathbb{R}^{3\times 3}$. We can prove this the following way:

$$D_x^T D_x = \frac{4}{10}\begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} \frac{4}{10}\begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \end{bmatrix} = \frac{16}{100}\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} = \begin{bmatrix} \frac{16}{50} & \frac{16}{100} & 0 \\ \frac{16}{100} & \frac{16}{50} & \frac{16}{100} \\ 0 & \frac{16}{100} & \frac{16}{50} \end{bmatrix} = L_{xx}$$

which is the FD matrix of negative 1D Laplacians in the x direction. Similarly,

$$D_y^T D_y = \frac{4}{5}\begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} \frac{4}{5}\begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \end{bmatrix} = \frac{16}{25}\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} = \begin{bmatrix} \frac{32}{25} & \frac{-16}{25} & 0 \\ \frac{-16}{25} & \frac{32}{25} & \frac{-16}{25} \\ 0 & \frac{-16}{25} & \frac{32}{25} \end{bmatrix} = L_{yy}$$

which is the FD matrix of negative 1D Laplacians in the y direction. Introducing the identity matrices $I_x = I_y = I_3$, we can construct the sparse 2D Laplacian matrix A corresponding to the lexicographic order of unknowns as

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

We can prove this in the following way:

$$I_x, I_y \in \mathbb{R}^{m\times n}$$

$$L_x x, L_y y \in \mathbb{R}^{3\times 3}$$

$$I_y \bigotimes L_{xx}, L_{yy} \bigotimes I_x \in \mathbb{R}^{3m \times 3n}$$

$$A \in \mathbb{R}^{3m \times 3n}$$

$$A \in \mathbb{R}^{9 \times 9}$$

m=3, n=3

$$I_y \bigotimes L_{xx} + L_{yy} \bigotimes I_x$$

$$= \begin{bmatrix} \frac{16}{50} & \frac{-16}{100} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{-16}{100} & \frac{16}{50} & \frac{-16}{100} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{-16}{100} & \frac{16}{50} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{16}{50} & \frac{-16}{100} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{-16}{100} & \frac{16}{50} & \frac{-16}{100} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{-16}{100} & \frac{16}{50} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{16}{50} & \frac{-16}{100} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{-16}{100} & \frac{16}{50} & \frac{-16}{100} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{-16}{100} & \frac{16}{50} \end{bmatrix} + \begin{bmatrix} \frac{32}{25} & 0 & 0 & \frac{-16}{25} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{32}{25} & 0 & 0 & \frac{-16}{25} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{32}{25} & 0 & 0 & \frac{-16}{25} & 0 & 0 & 0 \\ \frac{-16}{25} & 0 & 0 & \frac{32}{25} & 0 & 0 & \frac{-16}{25} & 0 & 0 \\ 0 & \frac{-16}{25} & 0 & 0 & \frac{32}{25} & 0 & 0 & \frac{-16}{25} & 0 \\ 0 & 0 & \frac{-16}{25} & 0 & 0 & \frac{32}{25} & 0 & 0 & \frac{-16}{25} \\ 0 & 0 & 0 & \frac{-16}{25} & 0 & 0 & \frac{32}{25} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{-16}{25} & 0 & 0 & \frac{32}{25} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{-16}{25} & 0 & 0 & \frac{32}{25} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{8}{5} & \frac{-16}{100} & 0 & \frac{-16}{25} & 0 & 0 & 0 & 0 & 0 \\ \frac{-16}{100} & \frac{8}{5} & \frac{-16}{100} & 0 & \frac{-16}{25} & 0 & 0 & 0 & 0 \\ 0 & \frac{-16}{100} & \frac{8}{5} & 0 & 0 & \frac{-16}{25} & 0 & 0 & 0 \\ \frac{-16}{25} & 0 & 0 & \frac{8}{5} & \frac{-16}{100} & 0 & \frac{-16}{25} & 0 & 0 \\ 0 & \frac{-16}{25} & 0 & \frac{-16}{100} & \frac{8}{5} & \frac{-16}{100} & 0 & \frac{-16}{25} & 0 \\ 0 & 0 & \frac{-16}{25} & 0 & \frac{-16}{100} & \frac{8}{5} & 0 & 0 & \frac{-16}{25} \\ 0 & 0 & 0 & \frac{-16}{25} & 0 & 0 & \frac{8}{5} & \frac{-16}{100} & 0 \\ 0 & 0 & 0 & 0 & \frac{-16}{25} & 0 & \frac{-16}{100} & \frac{8}{5} & \frac{-16}{100} \\ 0 & 0 & 0 & 0 & 0 & \frac{-16}{25} & 0 & \frac{-16}{100} & \frac{8}{5} \end{bmatrix} = A$$

2.2 Implementation

1. System matrix assembly for a rectangular domain.

   (a) Import the required modules as follows:
   Done in Jupyter Notebook.

   (b) Define the problem parameters as:
   Done in Jupyter Notebook.

   (c) Use sp.diags() to create sparse matrices D_x and D_y. Employing A.transpose() to transpose a sparse matrix A and A.dot(B) to find the (sparse) product of sparse matrices A and B, create sparse negative 1D Laplacian matrices L_{xx} and L_{yy}. Use sp.eye() to create sparse identity matrices I_x and I_y, and the Kronecker prod- uct function sp.kron() to create the sparse negative 2D Lapla- cian matrix A. Visualize A with plt.spy() for $N_x = N_y = 4$. Insert figure in your report. Print the values of A and compare those to the previously derived matrix in Section 2.1 (3).
   Done in Jupyter Notebook. The printed values of A match those of the previously derived matrix in Section 2.1 (3).

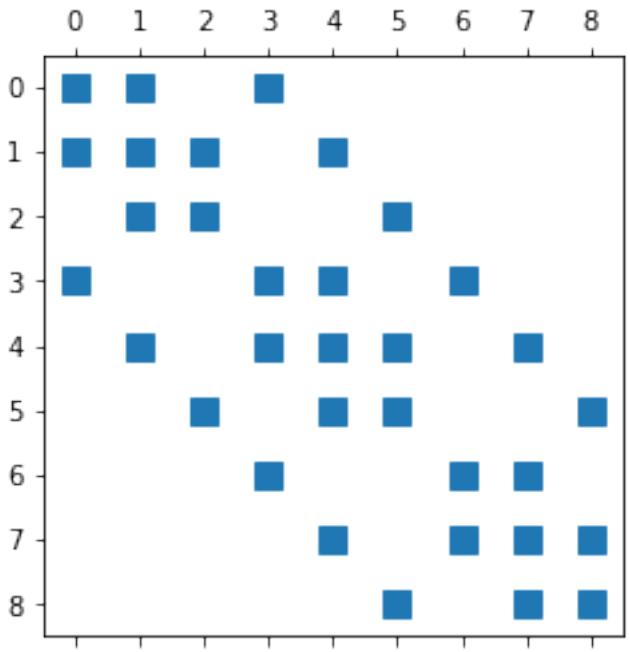

Abbildung 1: Plot from Matplotlib for 1. (c)

(d) Make a Python function that returns the sparse FD matrix of the negative 2D Laplacian with zero Dirichlet boundary conditions on a rectangular domain with a uniform lexicographic grid. Test you function with Nx = 200 and Ny = 200 (disable spy-plot for this).
Done in Jupyter Notebook.

2. Construction of the two-dimensional grid and evaluation of the source function on a rectangular domain.

(a) Create a Python function to evaluate the source function f (x, y) at any point. The function should work with scalar as well as array inputs.
Done in Jupyter Notebook.

(b) Create doubly uniform 2D grid on the rectangle Ω = (LeftX, RightX) $\times$ (LeftY, RightY), using the np.mgrid[...] class: Use Nx, Ny as grid control parameters (see help for np.mgrid[...]). Arrays x and y must contain only the inner points of the grid. Ex- plain the relation of the structure of x and y to the lexicographic ordering of the unknowns. Which array dimension corresponds to the x-direction and which to the y-direction along the coordinate axes?
Done in Jupyter Notebook. The array dimension for x and y are Nx $\times$ Nx and Ny $\times$ Ny respectively.

(c) Compute the grid values of the source function on rectangular domain with Nx = 200 and Ny = 100:
Done in Jupyter Notebook.

3. Lexicographic reshaping and solution of the algebraic problem.

(a) Visualize the source function as heat map using the function plt.imshow(). Make sure that:

- the plot axes show correct range for the values of x and y
- the orientation of the plot is correct, i.e., the x-axis is horizontal and the y-axis is vertical
- the y-axis points upwards

You can use the following template:
Insert the resulting figure in your report.
Done in Jupyter Notebook.

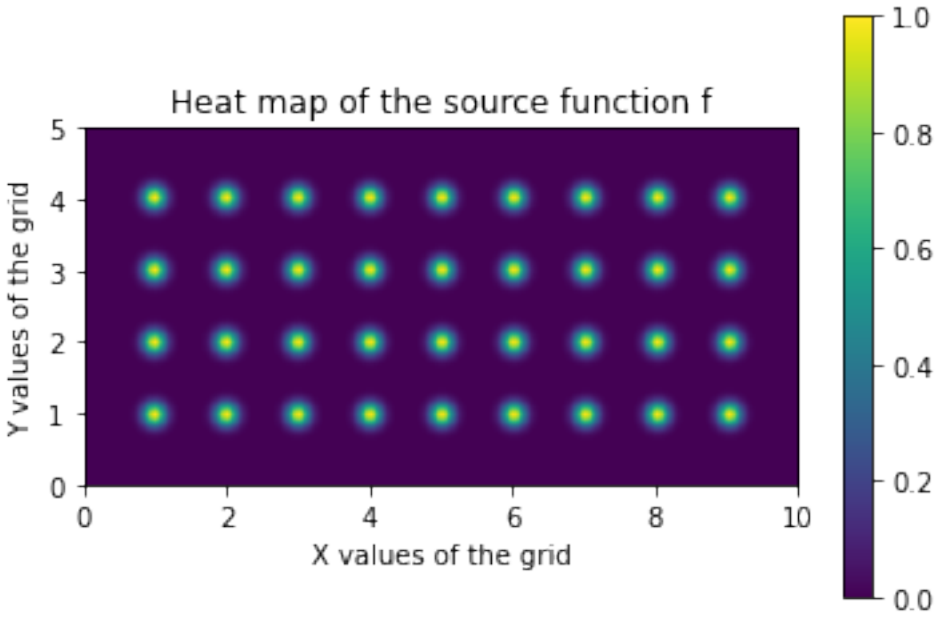

Abbildung 2: Plot from Matplotlib for 3. (a)

(b) Reshape the two-dimensional array f into the lexicographic RHS vector using the function np.reshape().
Done in Jupyter Notebook.

(c) Use previously created function to assemble the system matrix.
Done in Jupyter Notebook.

(d) Solve the linear system with the la.spsolve() function
What does the vector u contain?
Done in Jupyter Notebook. The vector u contains the values of the unknowns solved by linear algebra.

(e) Reshape the solution vector u into a two-dimensional array using the np.reshape() function and visualize the solution.
Insert the resulting image as a figure in your report.
Done in Jupyter Notebook.

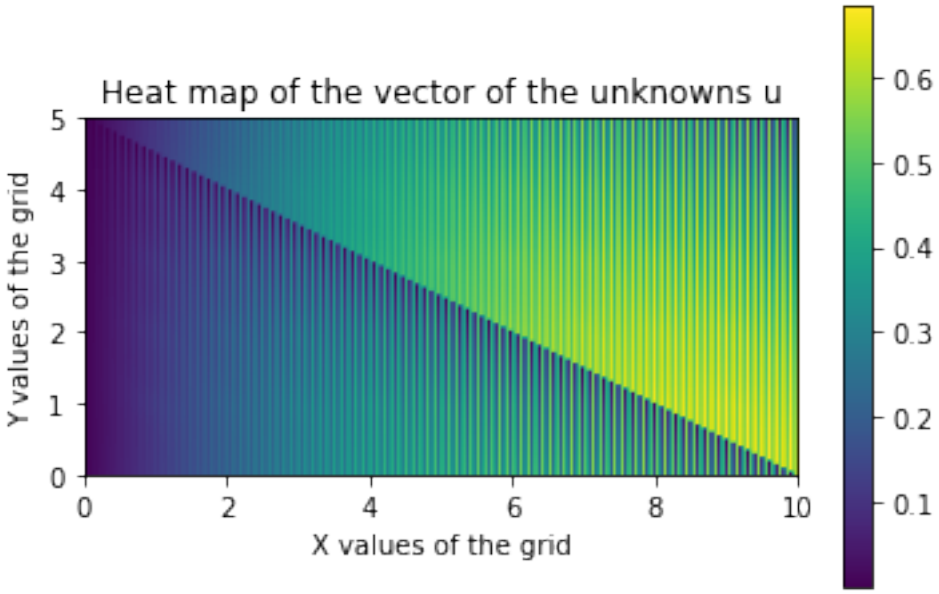

Abbildung 3: Plot from Matplotlib for 3. (e)

3 Finite-Volume Method

For this method we shall consider both the homogeneous coefficient function (2) and the following inhomogeneous coefficient function:

$$k(x, y) = 1 + 0.1(x + y + xy), (x, y) \in \overline{\Omega} \tag{4}$$

3.1 Discretization

Similar to the FD Method, we explicitly discretize the problem on a small mesh with 4 steps (sub-intervals) in both the x- and the y-directions and adopt the lexicographic ordering of the nodes and unknowns. However, the Finite-Volume (FV) discretization procedure should be written out for arbitrary k(x, y).

1. Apply the vertex-centered FV Method and write down all discrete FV equations of the problem for the lexicographic ordering of the unknowns. The FV equations should be averaged over the area of the elementary cell. You may keep the general notation $k_{i,j}, f_{i,j}$ for the grid values of the known functions, as well as h_x, h_y for the grid steps. However, the actual values of the indices i and j should be written out explicitly.
   The steady-state diffusion equation is as follows:

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

$$-k\frac{\partial u}{\partial \mathbf{n}} = \alpha(u - u_0), (x, y) \in \partial\Omega$$

   where $\Omega \subset \mathbb{R}^2$ is a two-dimensional domain; k(x,y) and f(x,y) are given functions; α and u_0 are given constants; and $\mathbf{n}$ is the outwards normal to the boundary $\partial\Omega$.
   The finite volume Ω_{ij} with the area V_{ij} is centered around the vertex (x_i, y_j) and has the boundary Γ_{ij}.
   Integrating over cell,

$$\iiint_{V_{ij}} [-\nabla \cdot (k\nabla u)]\, dV$$

   Applying the 2D divergence theorem,

$$\oint_{\Gamma_{ij}} (-k\nabla u \cdot \mathbf{n})\, d\Gamma = \iiint_{V_{ij}} f\, dV$$

$$\oint_{\Gamma_{ij}} (-k\frac{\partial u}{\partial \mathbf{n}})\, d\Gamma = \iiint_{V_{ij}} f\, dV$$

   Single-point approximation for the right-hand side,

$$\iiint_{V_{ij}} f\, dV \approx f_{ij} h_x h_y$$

   Splitting boundary in segments

$$\oint_{\Gamma_{ij}} (-k\frac{\partial u}{\partial \mathbf{n}})\, d\Gamma = \int_{\Gamma_W} (-k(-\frac{\partial u}{\partial x}))\, dy + \int_{\Gamma_E} (-k(\frac{\partial u}{\partial x}))\, dy + \int_{\Gamma_S} (-k(-\frac{\partial u}{\partial y}))\, dx + \int_{\Gamma_N} (-k(-\frac{\partial u}{\partial y}))\, dx$$

   Central-difference and single-point approximation for fluxes:
   Approximation of the integral brings you to an edge of the main grid. FD approximation of the partial derivative connects you with the vertex values of the unknown function.

$$\int_{\Gamma_W} (k(\frac{\partial u}{\partial x}))\, dy \approx h_y k \frac{\partial u}{\partial x}|_{(x_{i-1/2}, y_j)} \approx h_y k_{i-1/2,j} \frac{u_{i,j} - u_{i-1,j}}{h_x}$$

$$\int_{\Gamma_E} (-k(\frac{\partial u}{\partial x}))\, dy \approx -h_y k \frac{\partial u}{\partial x}|_{(x_{i+1/2}, y_j)} \approx -h_y k_{i+1/2,j} \frac{u_{i+1,j} - u_{i,j}}{h_x}$$

$$\int_{\Gamma_S} (k(\frac{\partial u}{\partial y}))\, dx \approx h_x k \frac{\partial u}{\partial y}|_{(x_i, y_{j-1/2})} \approx h_x k_{i,j-1/2} \frac{u_{i,j} - u_{i,j-1}}{h_y}$$

$$\int_{\Gamma_N} (-k(\frac{\partial u}{\partial y}))\, dx \approx -h_x k \frac{\partial u}{\partial y}|_{(x_i, y_{j+1/2})} \approx -h_x k_{i,j+1/2} \frac{u_{i,j+1} - u_{i,j}}{h_y}$$

Inner point FVM equation:

$$h_y k_{i-1/2,j} \frac{u_{i,j} - u_{i-1,j}}{h_x} - h_y k_{i+1/2,j} \frac{u_{i+1,j} - u_{i,j}}{h_x} + h_x k_{i,j-1/2} \frac{u_{i,j} - u_{i,j-1}}{h_y} - h_x k_{i,j+1/2} \frac{u_{i,j+1} - u_{i,j}}{h_y} = h_x h_y f_{ij}$$

Rearrange to recognize the system matrix entries and divide by $h_x h_y$ for proper averaging:

$$k_{i-1/2,j} \frac{u_{i,j} - u_{i-1,j}}{h_x^2} - k_{i+1/2,j} \frac{u_{i+1,j} - u_{i,j}}{h_x^2} + k_{i,j-1/2} \frac{u_{i,j} - u_{i,j-1}}{h_y^2} - k_{i,j+1/2} \frac{u_{i,j+1} - u_{i,j}}{h_y^2} = f_{ij}$$

i=0,1,....,4 j=0,1,....,4

2. For k(x, y) given by (2), write down the numerical values of all system matrix elements and compare the result to the FD system matrix obtained in Section 2.1 (3).
System matrix,

$$A = \begin{bmatrix} \frac{8}{5} & \frac{-16}{100} & 0 & \frac{-16}{25} & 0 & 0 & 0 & 0 & 0 \\ \frac{-16}{100} & \frac{8}{5} & \frac{-16}{100} & 0 & \frac{-16}{25} & 0 & 0 & 0 & 0 \\ 0 & \frac{-16}{100} & \frac{8}{5} & 0 & 0 & \frac{-16}{25} & 0 & 0 & 0 \\ \frac{-16}{25} & 0 & 0 & \frac{8}{5} & \frac{-16}{100} & 0 & \frac{-16}{25} & 0 & 0 \\ 0 & \frac{-16}{25} & 0 & \frac{-16}{100} & \frac{8}{5} & \frac{-16}{100} & 0 & \frac{-16}{25} & 0 \\ 0 & 0 & \frac{-16}{25} & 0 & \frac{-16}{100} & \frac{8}{5} & 0 & 0 & \frac{-16}{25} \\ 0 & 0 & 0 & \frac{-16}{25} & 0 & 0 & \frac{8}{5} & \frac{-16}{100} & 0 \\ 0 & 0 & 0 & 0 & \frac{-16}{25} & 0 & \frac{-16}{100} & \frac{8}{5} & \frac{-16}{100} \\ 0 & 0 & 0 & 0 & 0 & \frac{-16}{25} & 0 & \frac{-16}{100} & \frac{8}{5} \end{bmatrix}$$

The result is the same as the one we got for the FD system matrix obtained in Section 2.1 (3).

3.2 Implementation

1. System matrix and the RHS vector assembly. Here we study a different assembly method, common when programming in general (compiled) languages, such as C and FORTRAN. The method involves nested loops and a careful approach to array indexing.

   (a) The code should be a continuation of the FD code from Section 2.2. Define two Python functions that return the values of k(x, y) functions (2) and (4) for any input coordinates x and y:
   Visualize both coefficient functions on a 200 × 100 mesh in a single figure using the plt.subplot() commands and insert the corresponding figure in your report.
   Done in Jupyter Notebook.

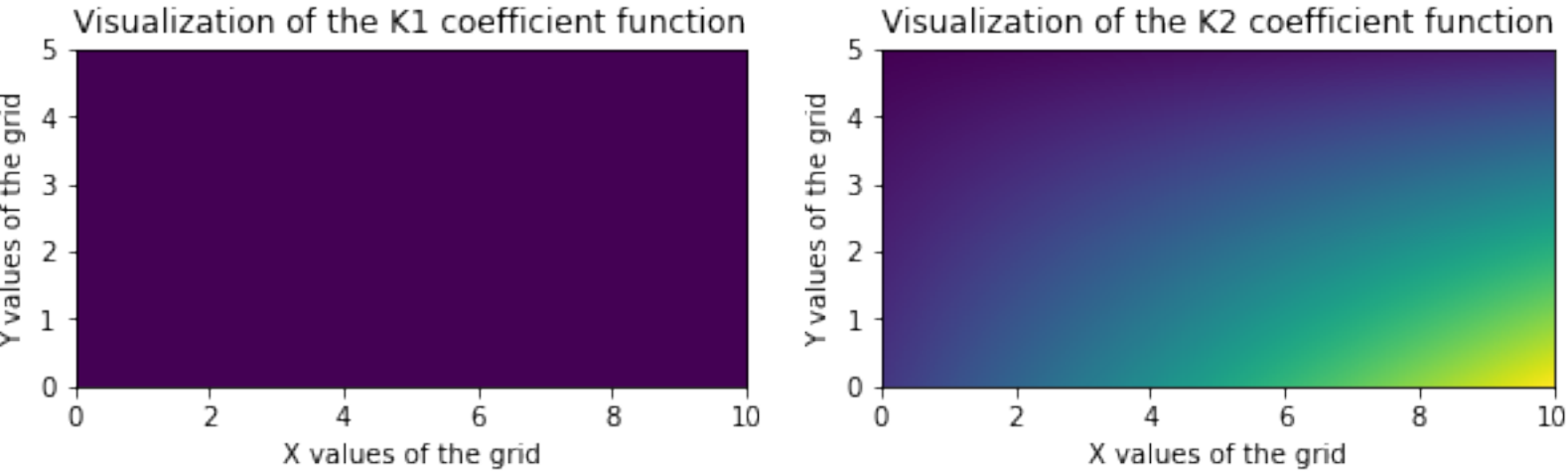

Abbildung 4: Plot from Matplotlib for 1. (a)

   (b) Use a (nested) loop over the grid points in lexicographic order to create one-dimensional numpy arrays of matrix diagonals. Then, construct the sparse system matrix from diagonals using sp.diags() with the additional option format='csc'. Wrap your code in a function that creates the 2D FVM matrix of the negative Laplacian with the coefficient function name passed as an argument.
   It should be possible to create the FVM system matrix A with the coefficient function k(x, y) given by (2) simply as follows:
   Done in Jupyter Notebook.

(c) Test your system matrix assembly code for Nx = 4, Ny = 4. Create the system matrix and compare it with your theoretical predictions in 3.1 (2). Create the system matrix with k(x, y) from (4) on this small grid and compare your matrix with the output of the example code of your TA.
Done in Jupyter Notebook. The system matrix obtained for k=1 matches my theoretical prediction from 3.1 (2).

(d) Create the RHS vector following the guidelines of Section 2.2.
Done in Jupyter Notebook.

2. Follow the guidelines and examples of Section 2.2 to find the numerical solutions on the grid with Nx = 200 and Ny = 100 for the coefficient functions (2) and (4). Plot the obtained numerical solutions with plt.imshow() and insert the corresponding figure in your report.

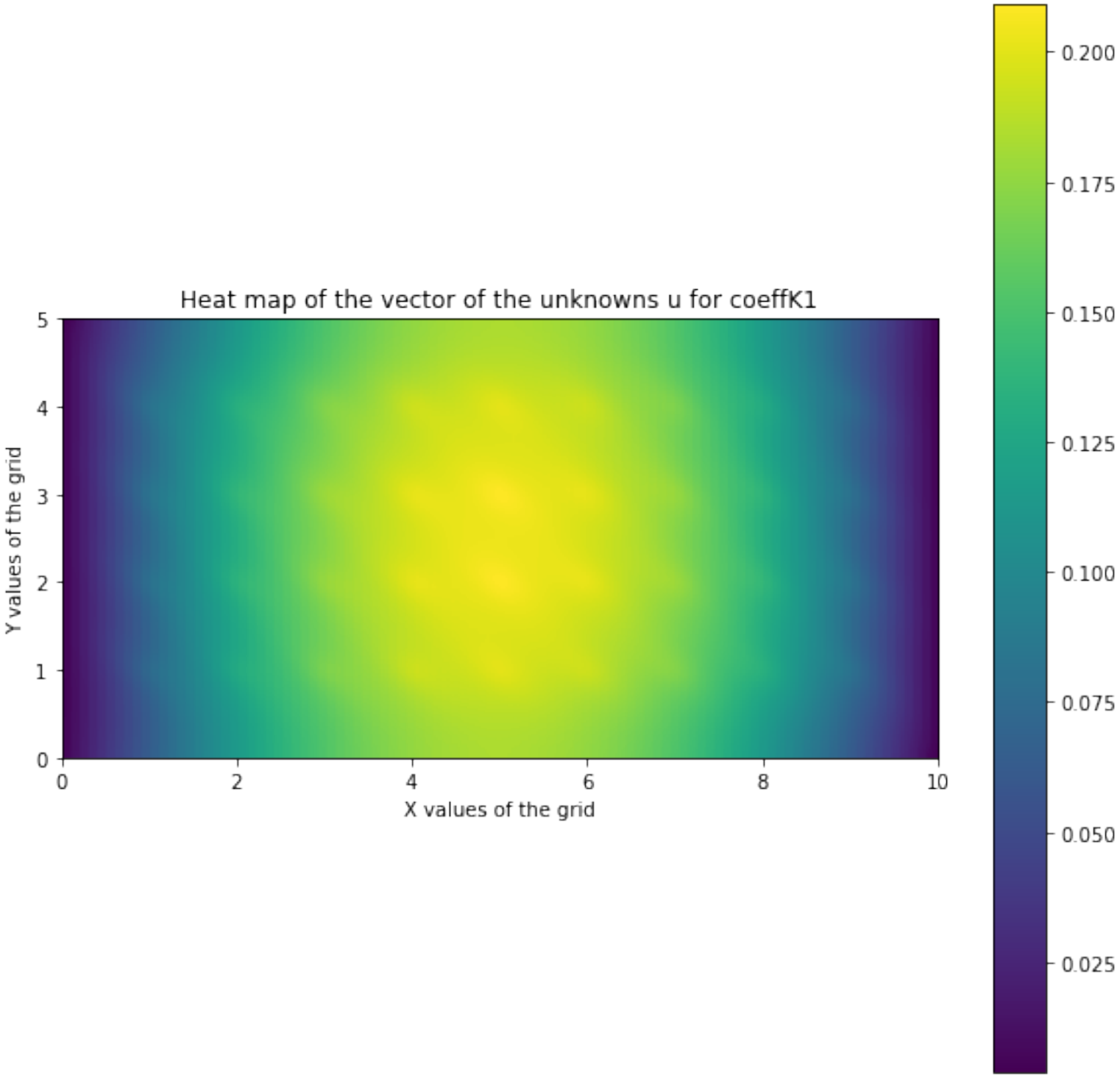

Abbildung 5: Plot from Matplotlib for 2. K1

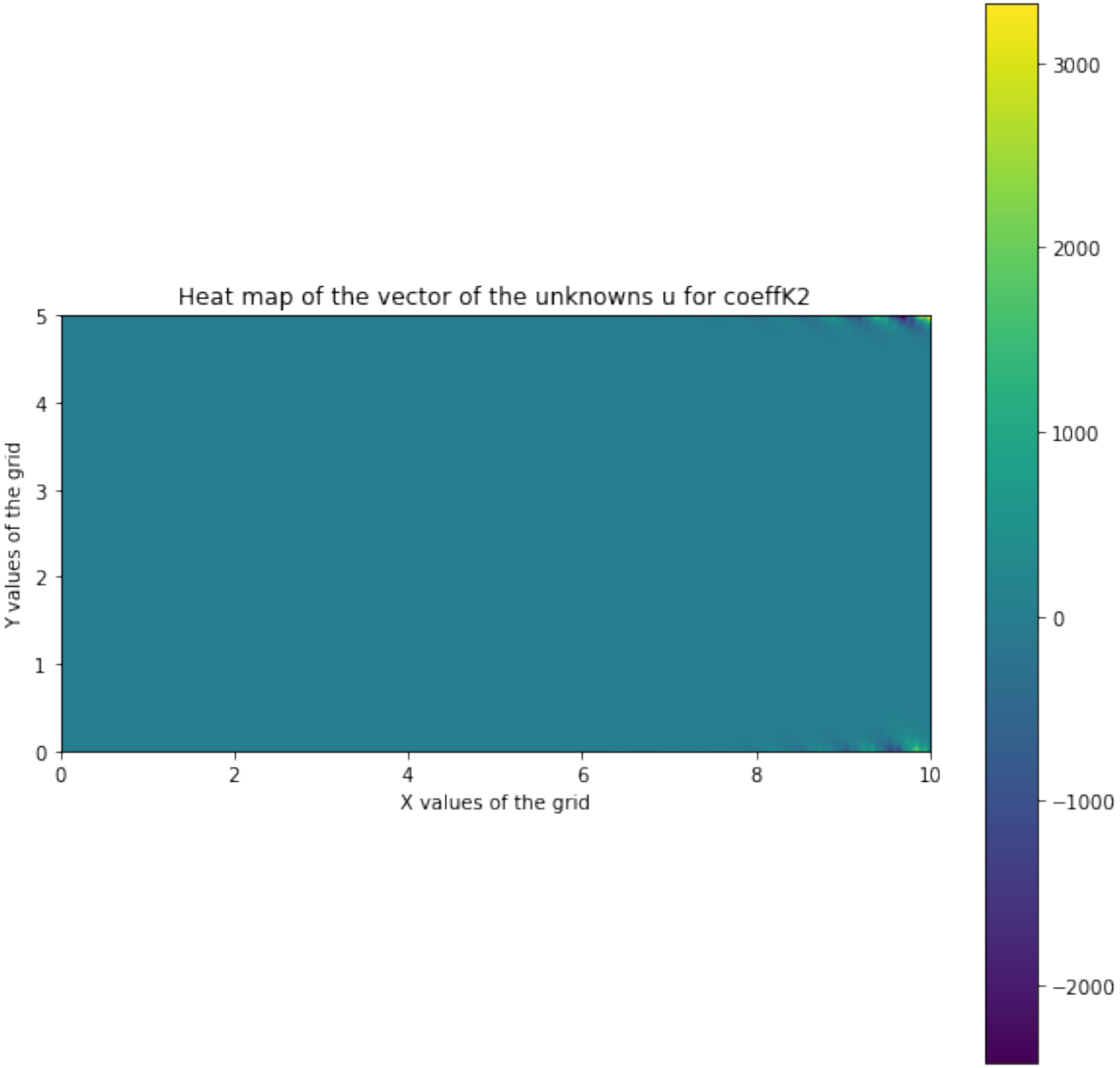

Abbildung 6: Plot from Matplotlib for 2. K2

4 Discussion

Explain why the obtained FDM and FVM solutions make sense from the physical and mathematical points of view.

The FDM and FVM solutions make sense because for k=1, the solution obtained from both methods is the same. For K2, the heat increases for higher values of x or higher values of y, as this increases the diffusion coefficient. Lower values of either/both K1 and K2, leads to a decrease in heat. Also, a high value of both x and y results in intermediate heat due to the 0.1xy term in the coefficient K2.