Assignment 3

Given Information
Consider the following boundary value problem for the Poisson equation:

$$-\Delta u = f,\ (x,y) \in \Omega$$

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

$$f(x,y) = 1 + x + y + xy,\ (x,y) \in \vec{\Omega} \tag{1}$$

$$Domain\,\Omega:\ (x,y) \in \Omega,\ \ if\ (x^2+y^2-1)^3 - x^2y^3 < 0$$

$$Boundary\,\partial\Omega:\ (x,y) \in \partial\Omega,\ \ if\ (x^2+y^2-1)^3 - x^2y^3 = 0$$

Problem 1
Finite-Difference discretization. First consider Ω to be a square domain (-1.5,1.5) × (-1.5,1.5). Divide this domain in 4 equal steps along both axes.

Solution
(a) As per the given information, $N_x = N_y = 4$. Grid steps $h_x = h_y = h = \frac{1.5-(-1.5)}{4} = 0.75$

(b) There are total 25 grid points for the grid step of 0.75 in both x and y axes. **9 points are internal grid points**, where the solution is to be computed and the remaining **16 points are boundary grid** points where we have Dirichlet boundary condition ($u = 0$).

(c) The $\mathcal{O}\left(h^2\right)$ approximation of $-\Delta u$ is:

$$-\Delta u = \frac{-u_{i-1,j} + 2u_{i,j} - u_{i+1,j}}{{h_x}^2} + \mathcal{O}\left(h_x^2\right) + \frac{-u_{i,j-1} + 2u_{i,j} - u_{i,j+1}}{{h_y}^2} + \mathcal{O}\left(h_y^2\right)$$

(d) Using the aforementioned equation with $h_x = h_y = h$:

$$-\Delta u = \frac{-(u_{i-1,j} + u_{i,j-1}) + 4u_{i,j} - (u_{i+1,j} + u_{i,j+1})}{h^2}$$

Upon solving for $i, j = 1, 2, 3$ and setting boundary point values as zero:

$$\begin{aligned}
4u_{1,1} - u_{2,1} - u_{1,2} &= f_{1,1} * h^2 \\
-u_{1,1} + 4u_{1,2} - u_{2,2} - u_{1,3} &= f_{1,2} * h^2 \\
-u_{1,2} + 4u_{1,3} - u_{2,3} &= f_{1,3} * h^2 \\
-u_{1,1} + 4u_{2,1} - u_{3,1} - u_{2,2} &= f_{2,1} * h^2 \\
-u_{1,2} - u_{2,1} + 4u_{2,2} - u_{3,2} - u_{2,3} &= f_{2,2} * h^2 \\
-u_{1,2} - u_{2,1} + 4u_{2,2} - u_{3,2} - u_{2,3} &= f_{2,3} * h^2 \\
-u_{2,1} + 4u_{3,1} - u_{3,2} &= f_{3,1} * h^2 \\
-u_{2,2} - u_{3,1} + 4u_{3,2} - u_{3,3} &= f_{3,2} * h^2 \\
-u_{2,3} - u_{3,2} + 4u_{3,3} &= f_{3,3} * h^2
\end{aligned}$$

(e) System Matrix for the above linear equations with lexicographic order is:

$$A/h^2 = \begin{bmatrix} 7.11111111 & -1.77777778 & 0 & -1.77777778 & 0 & 0 & 0 & 0 & 0 \\ -1.77777778 & 7.11111111 & -1.77777778 & 0 & -1.77777778 & 0 & 0 & 0 & 0 \\ 0 & -1.77777778 & 7.11111111 & 0 & 0 & -1.77777778 & 0 & 0 & 0 \\ -1.77777778 & 0 & 0 & 7.11111111 & -1.77777778 & 0 & -1.77777778 & 0 & 0 \\ 0 & -1.77777778 & 0 & -1.77777778 & 7.11111111 & -1.77777778 & 0 & -1.77777778 & 0 \\ 0 & 0 & -1.77777778 & 0 & -1.77777778 & 7.11111111 & 0 & 0 & -1.77777778 \\ 0 & 0 & 0 & -1.77777778 & 0 & 0 & 7.11111111 & -1.77777778 & 0 \\ 0 & 0 & 0 & 0 & -1.77777778 & 0 & -1.77777778 & 7.11111111 & -1.77777778 \\ 0 & 0 & 0 & 0 & 0 & -1.77777778 & 0 & -1.77777778 & 7.11111111 \end{bmatrix}$$

There are 33 non-zero elements in this system matrix.

(f) The system matrix can be obtained from the sparse first-order (partial) derivative matrices D_x and D_y, which are of the order $N_x \times N_x - 1$ and $N_y \times N_y - 1$ respectively. Here, $N_x = N_y = 4$:

$$D = D_x = D_y = \frac{1}{h}. \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \end{bmatrix}$$

The two sparse (partial) 1D Laplacian matrices are:

$$L = L_{xx} = L_{yy} = D^T D$$

$$\frac{1}{h}. \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} . \frac{1}{h}. \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \end{bmatrix} = \frac{1}{h^2}. \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \end{bmatrix}$$

Solving the above matrix multiplication, we get:

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

We can construct the sparse 2D Laplacian matrix A corresponding to the lexicographic order of unknowns using a kronecker product that we learnt in the lecture, with identity matrices I_x and I_y of the order $N_x - 1$ and $N_y - 1$ respectively:

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

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

$$A = \frac{1}{h^2}. \begin{bmatrix} 2.I & -1.I & 0.I \\ -1.I & 2.I & -1.I \\ 0.I & -1.I & 2.I \end{bmatrix} + \frac{1}{h^2}. \begin{bmatrix} 1.L & 0.L & 0.L \\ 0.L & 1.L & 0.L \\ 0.L & 0.L & 1.L \end{bmatrix}$$

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

$$A = \frac{1}{h^2}. \begin{bmatrix} 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 \end{bmatrix}$$

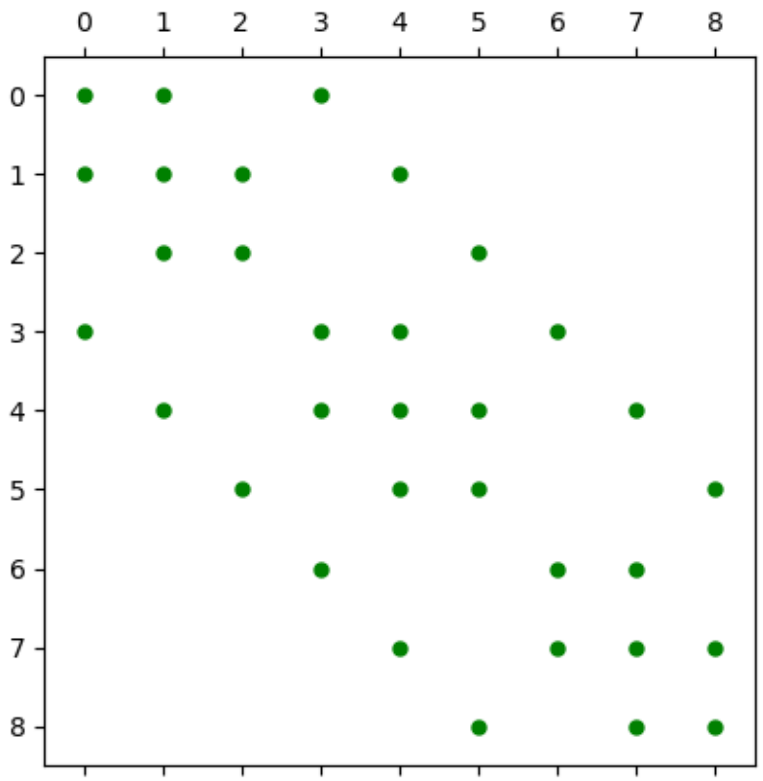
Substituting the value of h and multiplying with the matrix elements, we get the same system matrix as above.

(g) When the domain is deformed inwards from the rectangular domain, we are excluding the points outside the boundary. Our system matrix, source function and vector of unknowns would not include the grid points outside the defined boundary. If a previously internal point becomes a boundary point, the value of the source function and the unknown at that grid point would remain unchanged, as the domain boundary equation would still be satisfied with a new boundary condition.

Problem 2
System matrix assembly for a rectangular domain.

Solution
Implementing the steps mentioned in (a), (b), (c):

Spy Plot of the system matrix A



Problem 3
Construction of the two-dimensional grid and evaluation of the source function on a rectangular domain.

Solution
After implementing the steps mentioned in (a) and (b), we have the grid values for the inner points defined. The way *numpy.mgrid* is it start it's index from the bottom left coordinate of the actual cartesian grid and finishes on the top right coordinate. Hence, while plotting, we need to transpose the x and y matrices in order to plot with the y-axis pointing upwards.

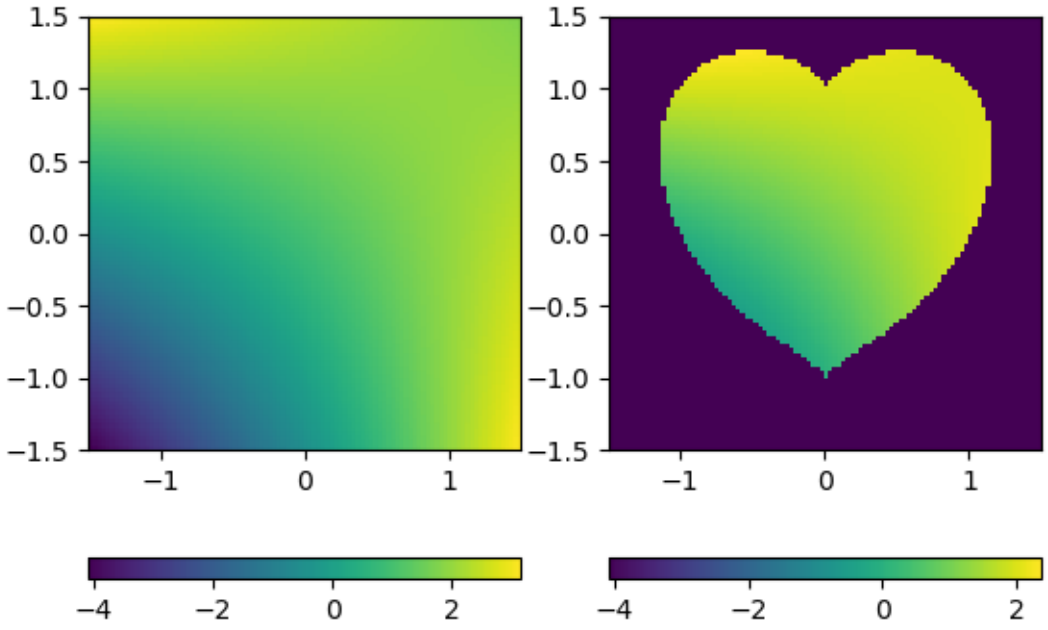
Lexicographically, taking the transpose works because instead of column-wise increments, we switch to row-wise increments in index. Reshaping matrices based on this grid gives us the required order.

The rows of the arrays correspond to the x-direction and the y-direction. The combination of the two arrays define every point in the grid based on our grid spacing.

Problem 4
Modification of the domain boundary. From this point onward we assume $N_x = 100$ and $N_y = 100$

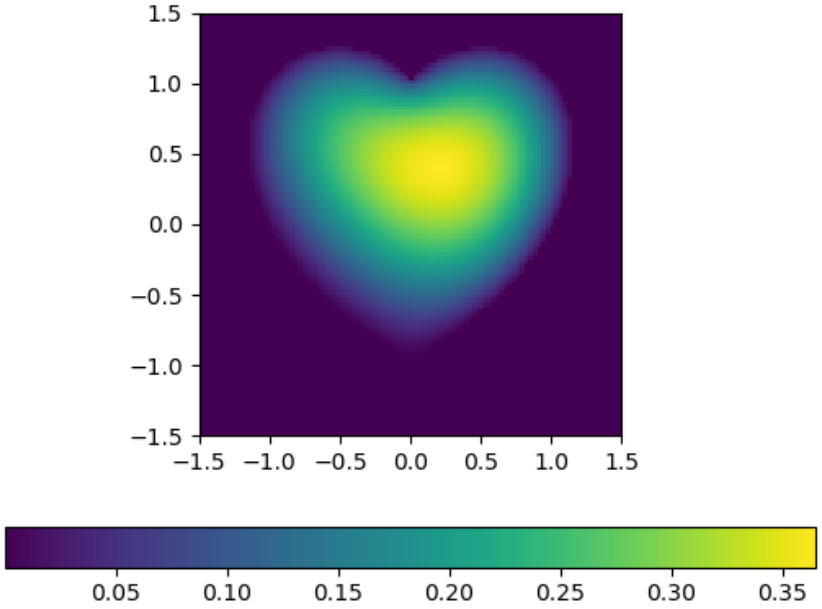
Solution
Implementing the steps mentioned in (a), (b), (c):

Source function f on Rectangular(L) and Given Domain(R)



Implementing the steps mentioned in (d), (e), (f), (g):

Source function f on Rectangular(L) and Given Domain(R)



(e) The last line in the code to define *Ad* truncates the A matrix to include only non-zero values to reduce the memory requirements. *tocsc()* and *tocsr()* also merges any duplicate values, so in the case of the deformed boundary, the inner nodes of the rectangular domain that satisfy the deformed boundary equations become the new boundary points like in 1(g).

The vector u contains the solutions at all the points inside the given domain. In the last line, we see that the rows and columns of *ufill* are populated with the values of the solution u at the right indices without mentioning it explicitly. This is because of how well python handles vectorization, the array elements are treated as vector components and we do not require loops or any other method for assigning array elements.

Problem 5
Eigenvalues and eigenvectors.

Solution
(a) Using the eigen-decomposed A, we can rewrite the linear equation:

$$A.u = f \implies u = A^{-1}.f$$
$$[V.\Lambda.V^T]^{-1}.f$$

Using $V^T = V^{-1}$; we get:

$$u = V.\Lambda^{-1}.V^T.f$$

(b) From the expression above, we can rewrite it as:

$$\begin{bmatrix} v_1 & v_2 & v_3 & . & . & . & v_n \end{bmatrix} . \begin{bmatrix} \frac{1}{\lambda_1} & \frac{1}{\lambda_2} & . & . & . & \frac{1}{\lambda_n} \end{bmatrix} . \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ . \\ . \\ . \\ v_n \end{bmatrix} .f$$

Where v_i is the i^{th} eigenvector, λ_i is the i^{th} eigenvalue, f is the source function vector and the "." here is an inner/dot product of the vectors. Writing in a general form of the i^{th} term:

$$v_i.\frac{1}{\lambda_i}.{v_i}^T.f$$

Summing over n would give us the value of the solution vector u:

$$u = \sum_{i=1}^{n} v_i.\frac{1}{\lambda_i}.{v_i}^T.f$$

Rearranging the terms, we get the value of α_i:

$$\alpha_i = \frac{1}{\lambda_i}.{v_i}^T.f$$

(c) The smallest eigenvalues and the associated eigenvectors would have the largest influence on the solution of u as the inverse of the value would make the terms the largest in the summation.

(e) After completing the computation in (d), we obtain the 20 smallest eigenvalues and eigenvectors of the system matrix. Out of these, the smallest eigenvalue would contribute to the previously obtained solution for u.

If the source function has been the 10th smallest eigenvector, our solution of u would look like:

Solution of u for Source function $f = v_{10}$