NTNU
Kunnskap for en bedre verden

Department of Mathematical Sciences

TMA4212 - Numerical solution of differential equations by difference methods

Project 1 - Heat distribution in anisotropic materials

Authors:
Karianne Strand Bergem, Kristin Fullu, Marius Bjerke Stjernstedt

25.02.2024

Table of Contents

List of Figures

1 Introduction

During this project we study numeric methods to help us solve a PDE. We develop and implement a finite difference scheme for elliptic 2-D problems in a specified domain. The project also tackles the issue of irregular grids. Further, we perform theoretical error analysis and aim to solve potential deficiencies for the scheme.

2 Heat distribution in anisotropic materials

This project will focus on a model of stationary temperature distribution T in a solid Ω given as

$$-\nabla \cdot (\kappa \nabla T) = f \qquad \text{in} \quad \Omega, \tag{1}$$

where κ is a constant for heat conductivity defined from the two directions of heat flow,

$$\vec{d_1} = (1, 0) \qquad \text{and} \quad \vec{d_2} = (1, r), \quad \text{where} \quad r \in \mathbb{R}.$$

The model can be rewritten as the following PDE,

$$-\nabla \cdot (\kappa \nabla T) = (a+1)\partial_x^2 T + 2r\partial_x \partial_y T + r^2 \partial_y^2 T = -a\partial_x^2 T - (\vec{d_2} \cdot \nabla)^2 T,$$

where a is a strictly positive coefficient. The domain Ω is defined by the $[0, 1] \times [0, 1]$ unit square and with Dirichlet boundary conditions $T = g$ on $\partial\Omega$.

2.1 Discretisation

First, we discretise the given problem for numerical solving. Using central differences for the second derivatives, we obtain the numerical scheme

$$-\mathcal{L}_h T_P = a\frac{-T_W + 2T_P - T_E}{h^2} + \frac{-T_{N'} + 2T_P - T_{S'}}{h^2}. \tag{2}$$

Here, P denotes the point in question, while W and E respectively denotes the points to the left and right of P. Similarly, N' denotes a point "above" P, and S' is a point "below" P. These have a prime to specify that they are not exactly one grid point above when considering an equally spaced grid, but skewed wrt. the $\vec{d_2}$ direction. Figure 1 visually represents the scheme.

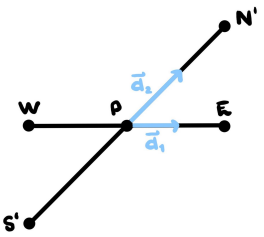

Figure 1: Stencil for the scheme

In our scheme, we have used the following ordering for the grid.

$$\begin{bmatrix} M^2 - 3M + 3 & \dots & \dots & (M-1)^2 \\ \vdots & \vdots & \ddots & \vdots \\ M & M+1 & \dots & 2(M-1) \\ 1 & 2 & \dots & M-1 \end{bmatrix}$$

Here, $M-1$ is the number of internal grid points in both x- and y-direction, meaning $M+1$ is the number of grid points in each direction when including the boundary. Because of 0-indexing in Python, the numeric values are all shifted accordingly. Combining this ordering and our stencil, appendix A presents an example for the boundary values needed for different points in a 3×3 matrix, available for a clearer understanding of the problem.

We can rewrite the scheme as a linear system,

$$B_h \vec{T} = \frac{1}{h^2}\vec{g} + \vec{f}, \quad \text{where} \quad B_h = \frac{1}{h^2}\begin{bmatrix} A & H & 0 \\ D & A & H \\ 0 & D & A \end{bmatrix}$$

is a block diagonal matrix with $(M-1) \times (M-1)$ tridiagonal matrices $A = \text{tridiag}\{-a, 2a+2, -a\}$, $H = \text{tridiag}\{0, 0, -1\}$, and $D = \text{tridiag}\{-1, 0, 0\}$. The $\vec{T}$ contains all the points T_P, the $\vec{g}$ contains the corresponding boundary points and $\vec{f}$ contains the corresponding f.

To test our method, we choose some simple functions to base our f and g on. We test with

$$u_1(x, y) = \sin(\pi x)\mathrm{e}^{-y} \text{ and } u_2(x, y) = \sin(\pi x) + \cos(2\pi y).$$

For the first choice, $g = u_1$ and $f = -a\partial_x^2 u_1 - (\vec{d_2} \cdot \nabla)^2 u_1$, and for the second choice, $g = u_2$ and $f = -a\partial_x^2 u_2 - (\vec{d_2} \cdot \nabla)^2 u_2$. Figure 2 shows the plots of the numerical and exact solutions for the two choices.

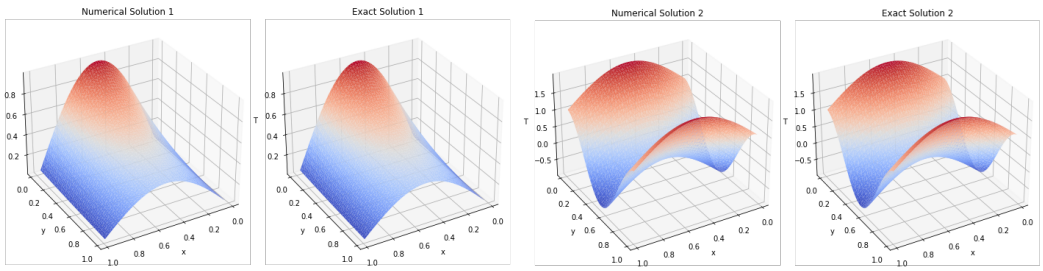

Figure 2: Numerical and exact solutions of u_1 and u_2.

For both choices, we can see that the numerical solution resembles the exact solution. This means that the scheme approximate the solutions of the functions well.

2.2 Stability

Further, we want to show that the scheme is L^∞ stable. This is a step in the process for finding a complete error bound. To show this stability we use the discrete maximum principle. In order to use it, we have to show monotonicity [1]. First we show that the scheme has positive coefficients considering the following equation

$$-\mathcal{L}_h T_P = \alpha_{PP} T_P - \sum_{Q \neq P} \alpha_{PQ} T_Q,$$

with coefficients

$$\alpha_{PP} = \frac{2a+2}{h^2} \text{ and } \alpha_{PQ} = \begin{cases} \frac{a}{h^2} & Q = W, E \\ \frac{1}{h^2} & Q = S', N' \\ 0 & \text{else.} \end{cases}$$

We have that $\alpha_{PP} \geq 0$ since $a > 0$. Further, we have that

$$\frac{2a+2}{h^2} = \alpha_{PP} \geq \sum_{Q \neq P} \alpha_{PQ} = 2\frac{a}{h^2} + 2\frac{1}{h^2} = \frac{2a+2}{h^2}.$$

Then, since $\alpha_{PP} \geq 0$ and $\alpha_{PP} \geq \sum_{Q \neq P} \alpha_{PQ}$, the scheme has positive coefficients and is monotone. Thus we can use the discrete maximum principle along with the comparison function $\phi = \frac{1}{2}x(1-x)$ to show L^∞-stability. We follow the method explained in week 6 of our course [2], beginning with inserting the comparison function in the scheme.

$$-\mathcal{L}_h \phi_P = -(a+1)\delta_x^2 \phi_P = a + 1 \geq 1 \quad (\text{as } \phi \geq 0 \text{ on } \Omega). \tag{3}$$

Further, we define $V_P := T_P - \|\vec{f}\|_\infty \phi_P$. Using equation (3), we have that

$$-\mathcal{L}_h V_P = -\mathcal{L}_h T_P - \|\vec{f}\|_\infty (-\mathcal{L}_h \phi_P) \leq f_P - \|\vec{f}\|_\infty \leq 0.$$

Since $-\mathcal{L}_h V_P \leq 0$, we can again use the discrete maximum principle. We use $\max_{\partial\mathbb{G}} T_P = 0$ to show that

$$V_P \leq \max_{\partial\mathbb{G}}\{0, V_P\} = \max_{\partial\mathbb{G}}\{0, T_P - \|\vec{f}\|_\infty \phi_P\} \leq \max_{\partial\mathbb{G}}\{0, T_P\} = 0.$$

Further, we use the definition of V_P and $V_P \leq 0$ to show that

$$T_P \leq \max_{\partial\mathbb{G}} |V_P| + \|\vec{f}\|_\infty \max_{\mathbb{G}} \phi_P \leq \|\vec{f}\|_\infty \max_{\mathbb{G}} \phi_P.$$

This can be done similarly for $-T_P$,

$$-T_P \leq \|-\vec{f}\|_\infty \max_{\mathbb{G}} \phi_P.$$

As the maximum of ϕ_P is $\phi_P(\frac{1}{2}) = \frac{1}{8}$, we can conclude that

$$\max_{\mathbb{G}} |T_P| \leq \|\vec{f}\|_\infty \max_{\mathbb{G}} \phi_P \leq \frac{1}{8}\|\vec{f}\|_\infty.$$

Thus we have shown that the scheme is L^∞-stable with stability constant $C = \frac{1}{8}$.

2.3 Error bound and convergence

Further, we want to derive an error bound for the scheme as follows

$$\|\vec{e}\|_\infty \leq C\|\vec{\tau}_h\|_\infty = \frac{1}{8}\|\vec{\tau}_h\|_\infty,$$

where $\vec{\tau}_h$ is the truncation error. To find this truncation error we use the central differences from exercise 1 and we can write the scheme as

$$-a(\delta_x^2 T + \frac{\delta_x^4 T}{12}h^2) - (\delta_{d_2}^2 T + \frac{\delta_{d_2}^4 T}{12}h^2) - \vec{f}_P = 0.$$

Now we write the truncation errors together to find the total truncation error,

$$-a\delta_x^2 T - \delta_{d_2}^2 T - a\frac{\delta_x^4 T}{12}h^2 - \frac{\delta_{d_2}^4 T}{12}h^2 - \vec{f}_P = 0.$$

Thus the truncation error is given by

$$\vec{\tau}_h = -\frac{a\delta_x^4 T + \delta_{d_2}^4 T}{12}h^2.$$

Further, we have that

$$\|\vec{\tau}_h\|_\infty = \max_m |\tau_m| \leq \frac{1}{12} K h^2$$

with $K = \|a\delta_x^4 T\|_\infty + \|\delta_{d_2}^4 T\|_\infty$. Then we have an error bound as follows

$$\|\vec{e}\|_\infty \leq \frac{1}{96} K h^2.$$

Thus we have shown convergence with a 2^{nd} order rate of convergence, $O(h^2)$, for smooth solutions.

2.4 Error and convergence analysis

In order to verify that our scheme is implemented correctly, a good option is to study the convergence rate. If the rate found numerically matches the value computed analytically, it is likely that code is correct. Finding the convergence rate is done through gradually decreasing the step size. Plotting with a polynomial fit in figure 3 shows us that for both functions, we obtain a 2^{nd} order rate of convergence. The exact values for step size, error and rate are presented in the tables in the code.

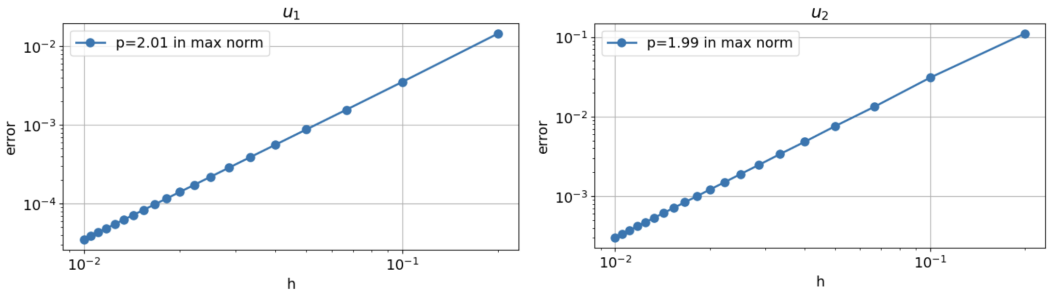

Figure 3: Numerical and exact solutions of u_1 and u_2, initial M=10.

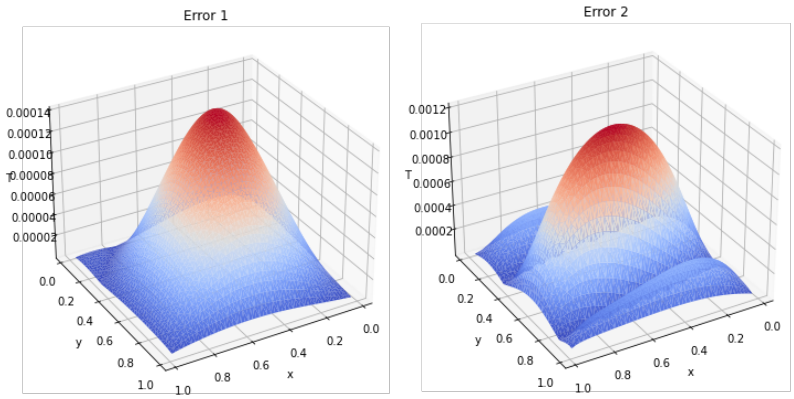

Figure 4: Error plots for u_1 and u_2, M = 50.

In figure 4 we observe that as the method moves away from the boundary, the error increases. This is expected, as the exact values are only given on the boundary. The errors however appear to be quite small, with small differences between u_1 and u_2.

2.5 Irrational step size in unit square

We now implement a new grid with step size $h = \frac{1}{M}$ and step size $k = |r|h$ for x- and y-direction respectively. Letting r be an irrational number, we see that in the last $M-1$ internal grid points, the point N' will overshoot the upper bound of the domain Ω.

To solve this issue, we have two options. Firstly we can choose to scale the stencil in this point, so that the point N' actually hits the boundary.

We instead chose to use option two, which is to fatten the boundary to include points that overshoot. This is a good alternative when values exists outside boundary, like for a continuous function. The way we implemented this approximates the boundary value with the next available point outside the boundary. Figure 5 therefore includes the point above y= 1. We found this to be an adequate approximation, because the step size is quite small. It is worth to note that this would not be possible with a much bigger r, because the value of the overshooting point would be too large.

However, we originally experimented more with this, and tried projecting this value down again onto the boundary. This method instead gave us a very irregular convergence, and a convergence order that differed between our two test functions. We therefore chose to omit this projection, and obtained the 2$^{\text{nd}}$ order convergence rate presented in figure 6.

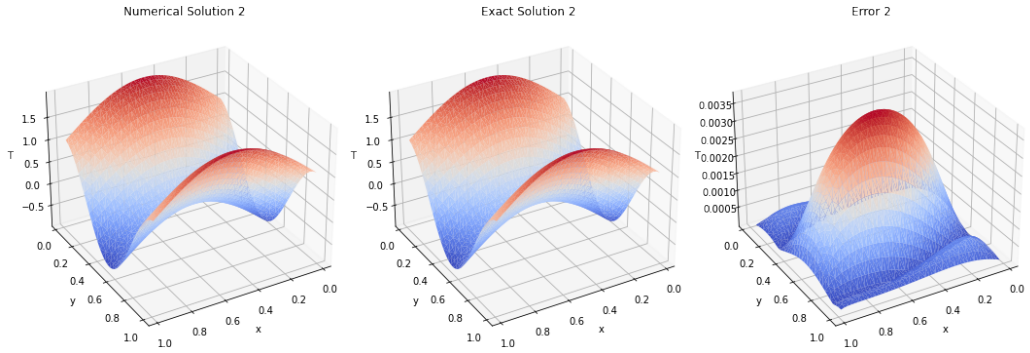

Figure 5: Numerical, exact solution, and error plot for u_2, M = 50 with irrational step size.

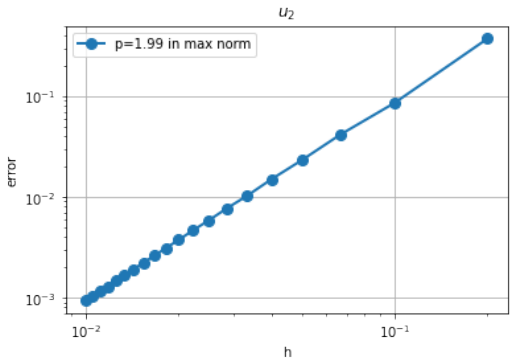

Figure 6: Convergence rate for u_2, M = 50 with irrational step size.

3 Dirichlet boundary value problem for irregular domain

Until now we have only considered the unit square domain for our method. In this task we will consider a new domain Ω, enclosed by the x- and y-axis together with the function $h(x) := \frac{1}{2}(\cos(\pi x) + 1)$. This is visualized in appendix B. Thus we can define $\partial\Omega$ as the curves

$$\gamma_1 = [0,1] \times \{0\}, \qquad \gamma_2 = \{0\} \times [0,1], \qquad \gamma_3 = \{(x, h(x)) : x \in [0,1]\}$$

We now consider the isotropic case for equation (1),

$$-\nabla \cdot (\kappa \nabla T) = -\nabla^2 T = -\Delta T = f$$

and implement two schemes to correct the problem with the stencil surpassing the boundary of the domain. The first is to fatten the boundary and the second is to modify the stencil near the boundary of the domain. For both implementations we will consider a uniform grid i.e. step sizes in both x and y direction are equal.

3.1 Fattening of the boundary

By fattening the boundary we can approximate the values of the boundary points where the stencil exceeds the domain. While we previously tried to implement a version of the fattening method that utilised projection to the nearest defined boundary point, it did not yield proper results, and we will therefore utilise the simplified here as well. Here, we must also remember that the function used for the boundary conditions must be viable in the rest of the 1st quadrant of $\mathbb{R}^2$.

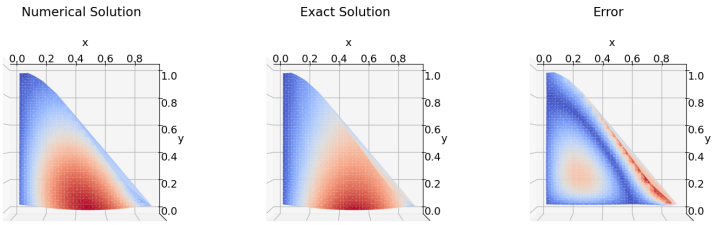

Figure 7: Numerical, exact solution, and error in 2-D with fattened boundary for u_1

In the error plot in figure 7 we observe a slight error in the middle of the boundary, corresponding to our thought earlier that the further you are from the boundary, the greater the probability of error. We also notice that the right sloped side of the boundary has a greater error, consistent with this being the boundary in need of fattening. From the plot of the numerical solution, we see that the value here is too low in comparison to the exact solution.

3.2 Modifying the stencil near the boundary

For the points where the stencil will overshoot the boundary we have to scale it such that we get accurate boundary points on the right hand side. If we consider a point P where the point E overshoots, we can in the scheme scale the step size such that a point $E' = P + (\eta_1 h, 0)$, where $\eta_1, \eta_2 \in (0, 1)$. By looking at the x- and y-direction separately we can derive the constants by Taylor expansion and central differences,

$$\partial_x^2 u_p - \tau_{x,p} = a_W u_W + a_P u_P + a_{E'} u_{E'},$$

where $\tau_{x,p}$ is the error term form simplifying the x-derivative. By Taylor expansion we get,

$$\begin{aligned}
\partial_x^2 u_p - \tau_{x,p} &= a_W u(x_P - h, y_P) + a_P u(x_P, y_P) + a_{E'} u_(x_P + \eta_1 h, y_P) \\
&= a_W \Big(u - h u_x + \frac{1}{2} h^2 u_{xx} + O(h^3) \Big) + a_P u(x_P, y_P) \\
&\quad + a_{E'} \Big(u + \eta_1 h u_x + \frac{1}{2} (\eta_1 h)^2 u_{xx} + O((\eta_1 h)^3) \Big).
\end{aligned}$$

We can now use that we only want the second derivative. Thus we get the set of equations,

$$\begin{cases} a_W + a_P + a_{E'} = 0 \\ \eta_1 h a_{E'} - h a_W = 0 \\ \frac{1}{2}(\eta_1 h)^2 a_{E'} + \frac{1}{2} h^2 a_W = 1 \end{cases} \quad => \quad \begin{cases} a_W + a_P + a_{E'} = 0 \\ \eta_1 a_{E'} = a_W \\ \eta_1^2 a_{E'} + a_W = \frac{2}{h^2} \end{cases}$$

Inserting $\eta_1 a_{E'}$ for a_W in the last equation we can solve for $a_{E'}$ and from there insert and calculate both a_W and a_P,

$$a_{E'} = \frac{2}{h^2 \eta_1 (\eta_1 + 1)}, \qquad a_W = \frac{2}{h^2 (\eta_1 + 1)}, \qquad a_P = -\frac{2}{h^2 \eta_1}.$$

We now do the same for the y-direction, with $N' = P + (0, \eta_2 k)$, and end up with similar constants,

$$b_{N'} = \frac{2}{k^2 \eta_2 (\eta_2 + 1)}, \qquad b_S = \frac{2}{k^2 (\eta_2 + 1)}, \qquad b_P = -\frac{2}{k^2 \eta_2}.$$

Our implementation uses a uniform grid ($k = h$) and thus the error term is $\tau = O(h)$. We know we have 3 cases for the modified stencil, i) point E exceeds the boundary in x-direction, ii) point N exceeds the boundary in y

direction, iii) both point N and E exceeds the boundary. Thus we can define a method that is utilised near the boundary,

$$-\tilde{\mathcal{L}}_h U_P = -a_{E'} U_{E'} - a_W U_W - b_{N'} U_{N'} - b_S U_S + (a_P + b_P) U_P.$$

We see how the scheme will change depending on if it is exceeding the left or the upper boundary, or if it is exceeding both. In the cases where only one is overshooting the corresponding η_i will be equal to 1.

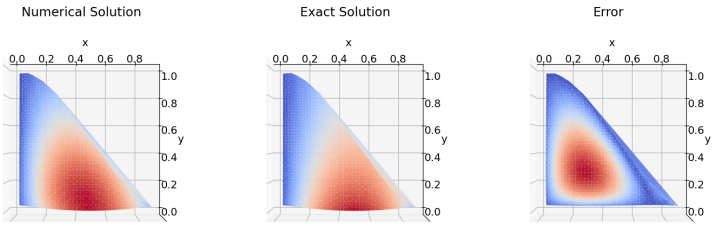

Figure 8: Numerical, exact solution, and error in 2-D with modified stencil for u_1

In figure 8 we observe a smaller error on the right boundary than with the fattening boundary. This corresponds with the fact that we actually adjust the scheme to hit the boundary points, meaning the right values are evaluated. As before, the error increases as you move away from the boundary.

3.3 Comparison of methods for irregular boundaries

When comparing the implementation of the two methods, one can conclude that in our case, the scheme for fattening the boundary is the easiest. This is because there is no need to take into account some scaling factor that varies from point to point. However, if the fattening scheme had used projection as we attempted, the stencil modifying method could have been preferred and viewed as easier as a projection can be more difficult to implement in our case.

On another note, when considering the errors of both methods, it is more difficult to find a clear choice as to which method is preferred. This all depends on where the exactness of the method is most important, as the fattening method had a better overall approximation, but the stencil modifying method has a better approximation of the exact boundary.

4 Conclusion

The scheme we derived in equation (2) approximated the solutions of our two chosen functions well with error that appear to be quite small. The rate of convergence found numerically matched the 2^{nd} order rate computed analytically, which verifies that the scheme was implemented correctly. For quite small irrational step sizes, we found it adequate to fatten the boundary to include points that overshoot. This was confirmed by again obtaining a 2^{nd} order convergence rate. For our two methods for the case of irregular boundaries, we observed a smaller error at the boundary when we modified the stencil. However, when fattening the boundary we got a overall smaller error. In our case, the fattening of the boundary is the easiest method to implement.

References

[1] Jakobsen, E.R. 2024. Lecture notes week 2. *TMA4212 Numerisk løsning av differensialligninger med differansemetoder*. Available from: Lecture notes week 2. Retrieved 2024/02/25.

[2] Jakobsen, E.R. 2024. Lecture notes week 6. *TMA4212 Numerisk løsning av differensialligninger med differansemetoder*. Available from: Lecture notes week 6. Retrieved 2024/02/25.

[3] NTNU. 2024. Project 1. *TMA4212 Numerisk løsning av differensialligninger med differansemetoder*. Available from: Project 1. Retrieved 2024/02/25.

Appendix

A Example of a 9 point domain for task 1a

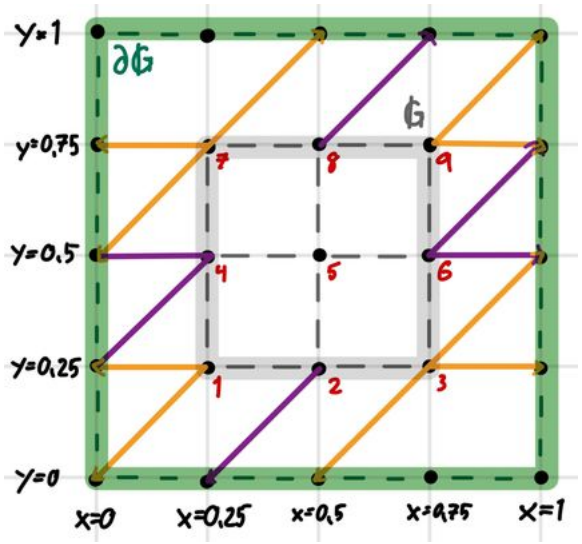

Figure 9: Example of the grid domain Ω for $M = 4$

B Domain to consider in problem with irregular domain

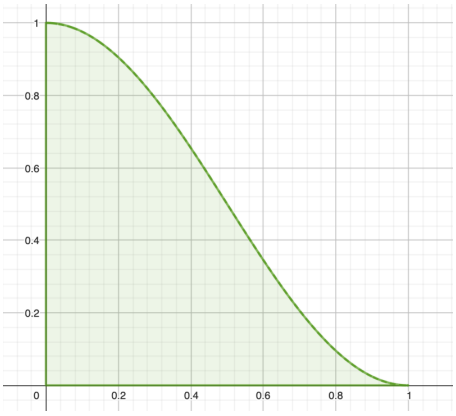

Figure 10: The domain Ω, from [3]