TMA4212
Discretizing, Modifying and Fattening - A Numerical Approach for Determining Staionary Heat Distributions

Eirik Fagerbakke, Henrik Grenersen and Halvard Sand-Larsen

February 2023

Contents

0 Introduction **1**

1 Problem 1 **1**
1.1 a) - Discretization and numerical solution of our problem 1
1.2 b) - Convergence results for the numerical scheme 2
1.2.1 Monotonicity 2
1.2.2 Local truncation error 3
1.2.3 Stability and error bound 3
1.3 c) - Numerical testing of the scheme 4
1.4 d) - Bad directions and complications 5

2 Problem 2 **5**
2.1 Fattening the boundary 6
2.2 Modifying our discretization 6
2.3 Results and comparison of the methods 6
2.3.1 Timing the methods 7

3 Conclusion **7**

4 Appendix **8**

0 Introduction

In this project we will look into modelling stationary heat distributions, first in anisotropic materials (Problem 1) and later in an irregular domain (Problem 2). Throughout the project we will consider the following model, given by Fourier's law:

$$-\nabla \cdot (\kappa \nabla u) = f \quad \text{in} \quad \Omega$$

Where $u(x, y)$ represents the temparature at the point (x, y) and κ is a matrix that represents the heat conductivity of the material. In order to work with this problem numerically, we will also discretize our domain Ω into a grid, denoted by $\mathbb{G}$.

1 Problem 1

In this problem we will consider a heat conductivity of the following form:

$$\kappa = \begin{bmatrix} a+1 & r \\ r & r^2 \end{bmatrix}$$

We can then reformulate our original problem as

$$-(a+1)\partial_x^2 u - 2r\partial_x\partial_y u - r^2\partial_y^2 u = -a\partial_x^2 u - (\mathbf{d}_2 \cdot \nabla)^2 u = f \qquad \mathbf{d}_2 = (1, r)^\top$$

We will also consider a rectangular domain $\Omega = [0, 1] \times [0, 2]$, and we now aim to derive a numerical scheme which we can utilize in order to determine the heat distribution in our domain, with Dirichlet boundary conditions.

1.1 a) - Discretization and numerical solution of our problem

In this task, we are working with a PDE of the form

$$-\mathcal{L}u = -ku_{xx} - 2bu_{xy} - cu_{yy} - du_x - eu_y - lu = f \qquad \in \Omega$$

We need to discretize this, i.e. find coefficients α_{PQ_i} such that

$$-\mathcal{L}u_p = -\mathcal{L}_h u_p + \tau_p$$
$$-\mathcal{L}U_p = \alpha_{pp}U_p - \sum_{i=1}^{s} \alpha_{PQ_i} U_{Q_i}$$

From the lectures and the note of Brynjulf Owren, we know that we can determine these coefficients by following the conditions (which follow from a Taylor expansion) given on page 72 in Owren [1]. We have chosen to split our differential equation into two different cases, by looking at

$$-\mathcal{L}_1 u = -a\partial_x^2 u \qquad -\mathcal{L}_2 u = -(\mathbf{d}_2 \cdot \nabla)^2 u$$

For $\mathcal{L}_1$, we use the usual central difference scheme, which is known as

$$-\mathcal{L}_1 u_m^n = -\frac{a}{h^2}\left(u_{m+1}^n - 2u_m^n + u_{m-1}^n\right) + O(h^2)$$

For $\mathcal{L}_2$, we also use a central difference scheme with points along the direction $\mathbf{d}_2$

$$-\mathcal{L}_{2_h} U_m^n = \alpha_0 U_m^n - \alpha_1 U_{m-1}^{n-1} - \alpha_2 U_{m+1}^{n+1}$$

Because we here discretize in both the x and y direction, we have to determine the above coefficients so that they fulfill the previously mentioned set of conditions. For our problem, we

have $k = 1, b = 2, c = 4, l = e = d = 0$. With $r = 2$ we get a step size of $\sqrt{5}h$. Using the conditions from page 72 in Owren we then get that our coefficients are $\alpha_0 = \frac{2}{h^2}, \alpha_1 = \alpha_2 = \frac{1}{h^2}$, so we have that

$$-\mathcal{L}_{2_h} U_m^n = -\frac{1}{h^2}\left(U_{m+1}^{n+1} - 2U_m^n + U_{m-1}^{n-1}\right)$$

In total, we have the scheme

$$-\mathcal{L}_h U_m^n = \frac{1}{h^2}\left(2(a+1)U_m^n - aU_{m+1}^n - aU_{m-1}^n - U_{m+1}^{n+1} - U_{m-1}^{n-1}\right)$$

To write this in matrix form, we first need to consider an ordering of the points. We have chosen to use a natural ordering, to translate the matrix U into a vector $\mathbf{U}$ consisting of the $(M-1)^2$ interior points. We then have

$$A\mathbf{U} = h^2\mathbf{f} + \mathbf{b}$$

with

$$B = \text{tridiag}\left\{-a, 2(a+1), -a\right\} \qquad C = \text{tridiag}\left\{0, 0, -1\right\} \qquad A = \text{tridiag}\left\{C^\top, B, C\right\}$$

We also have interior points which have points on their stencil that lie on the boundary, which we need to add to $\mathbf{b}$ on the right hand side. We have implemented this by array slicing, and filling in the values separately for the points on the lower, upper, right and left boundary. To test our implementation we have plotted some results for $a = 1$ and $a = 100$ which can be seen in figure 1. In the figure we see that increasing a gives an error that is approximately constant in y-direction. This is not the case for the x-direction, for which we observe an error that increases as we approach $x = 0.5$ from both sides, and we also note that there seems to be a greater area for which we make our maximal error in the case of the greater a. A possible explanation for this might be that the large a scales the derivative in the x-direction to such a great extent that it overpowers the variation caused by the derivative in the y-direction.

1.2 b) - Convergence results for the numerical scheme

1.2.1 Monotonicity

We would now like to determine if our scheme is monotone, as this would allow us to make use of the discrete maximum principle, which will be useful in the error analysis. For our scheme to be monotone, it first needs to be a method of positive coefficients. This is easily shown, as all coefficients are non-negative ($a > 0$). We further need to satisfy the "diagonal dominance" condition of the coefficients, that is

$$\alpha_{PP} \geq \sum_i \alpha_{PQ_i} \implies 2(a+1) \geq a + a + 1 + 1 = 2(a+1)$$

so we have a method of positive coeffcients. We also need our method to be boundary connected, which essentially means that we must be able to walk from any interior point on our grid to the boundary. This is easily verified from our stencil, which is illustrated in figure 2.

1.2.2 Local truncation error

To determine the local truncation error, we need to do a Taylor expansion. We have that

$$u(x_m \pm h, y_n) = u^n_{m\pm1} = u^n_m \pm h\partial_x u^n_m + \frac{h^2}{2}\partial_x^2 u^n_m \pm \frac{h^3}{6}\partial_x^3 u^n_m + \frac{h^4}{24}\partial_x^4 u^n_m \pm O(h^5)$$

$$\begin{aligned} u(x_m \pm h, y_n \pm 2h) = u^{n\pm1}_{m\pm1} = u^n_m &\pm h(\partial_x + 2\partial_y)u^n_m + \frac{h^2}{2}(\partial_x + 2\partial_y)^2 u^n_m \\ &\pm \frac{h^3}{6}(\partial_x + 2\partial_y)^3 u^n_m + \frac{h^4}{24}(\partial_x + 2\partial_y)^4 u^n_m \pm O(h^5) \end{aligned}$$

Plugging this into our scheme, we get

$$-\mathcal{L}_{1_h} = -\frac{a}{h^2}(u^n_{m+1} - 2u^n_m + u^n_{m-1}) = -a\partial_x^2 u^n_m - a\frac{h^2}{12}\partial_x^4 u^n_m + O(h^4)$$

$$-\mathcal{L}_{2_h} = -\frac{1}{h^2}(u^{n+1}_{m+1} - 2u^n_m + u^{n-1}_{m-1}) = -(\partial_x + 2\partial_y)^2 u^n_m - \frac{h^2}{12}(\partial_x + 2\partial_y)^4 u^n_m + O(h^4)$$

We then find the local truncation error as

$$\begin{aligned} \tau^n_m &= -\mathcal{L}u^n_m - (-\mathcal{L}_h u^n_m) = \frac{h^2}{12}(a\partial_x^4 + (\partial_x + 2\partial_y)^4)u^n_m + O(h^4) \\ &= \frac{h^2}{12}((a+1)\partial_x^4 + 8\partial_x^3\partial_y + 24\partial_x^2\partial_y^2 + 32\partial_x\partial_y^3 + 16\partial_y^4)u^n_m + O(h^4) \end{aligned}$$

Giving us a bound

$$\|\tau\| \leq Dh^2$$

With

$$D = \frac{1}{12}\left((a+1)\max_\Omega|\partial_x^4 u| + 8\max_\Omega|\partial_x^3\partial_y u| + 24\max_\Omega|\partial_x^2\partial_y^2 u| + 32|\partial_x\partial_y^3 u| + 16\max_\Omega|\partial_y^4 u|\right)$$

meaning that our method has order of consistency 2.

1.2.3 Stability and error bound

To investigate stability and get an error bound, we use the comparison function

$$\phi(x) = \frac{1}{2}x(1-x)$$

We first note that this function is nonnegative on our domain. Second, we also find that

$$-\mathcal{L}_h\phi(x) = \frac{1}{h^2}(2(a+1)\phi(x) - (a+1)\phi(x+h) - (a+1)\phi(x-h)) = a + 1 > 1$$

We now use the solution V of $-\mathcal{L}V = f$, with $V = 0$ on the boundary:

$$-\mathcal{L}_h(V_p - \|\mathbf{f}\|_\infty\phi_p) = f_p - \|\mathbf{f}\|_\infty(-\mathcal{L}_h\phi_p) < 0$$

We can now use the discrete maximum principle, which yields

$$V_p - \|\mathbf{f}\|_\infty\phi_p \leq \max_\Omega\{0, V_p - \|\mathbf{f}\|_\infty\phi_p\} = 0$$

We now get an upper bound on V_p as

$$V_p \leq \phi_p\|\mathbf{f}\|_\infty$$

By doing the same procedure for $(-V, -f)$, we get an equivalent lower bound:

$$-V_p \leq \phi_p \|-\mathbf{f}\|_\infty = \phi_p \|\mathbf{f}\|_\infty$$

Together, this yields

$$\max V_p \leq \max \phi_p \|\mathbf{f}\|_\infty = \frac{1}{8}\|\mathbf{f}\|_\infty$$

This also provides an error bound, as we have the relation

$$e_p = u_p - U_p \Rightarrow -\mathcal{L}_h e_p = -\tau_p$$

Since we also know the value u takes on the boundary, we can establish that $e_p = 0$ on $\partial\mathbb{G}$. We can then use the bound on V_p derived above, which yields the error bound

$$\|\mathbf{e}_h\|_\infty \leq \frac{1}{8}\|\tau_h\|_\infty = Ch^2$$

$$C = \frac{1}{8}D = \frac{1}{96}\left((a+1)\max_\Omega|\partial_x^4 u| + 8\max_\Omega|\partial_x^3\partial_y u| + 24\max_\Omega|\partial_x^2\partial_y^2 u| + 32|\partial_x\partial_y^3 u| + 16\max_\Omega|\partial_y^4 u|\right) \tag{1}$$

The rate of convergence is then found from the exponent of h as $p = 2$.

1.3 c) - Numerical testing of the scheme

Having done both some numerical and theoretical work, we now aim to test our scheme. Firstly, this is done for the previously mentioned test function $u(x, y) = \sin \pi x \cos 2\pi y$, which results in figure 4. The error bound for this particular problem was found from equation 1 as

$$\|\mathbf{e}_h\|_\infty \leq \frac{1}{96}\pi^4(a + 625)h^2 \overset{a=1}{=} \frac{313}{48}\pi^4 h^2$$

From the error plot we observe that the error attains minimal absolute values along the boundary, which we also expected, as we know the boundary values. However, the error also increases towards the centre of our domain, which seems reasonable, given that the values in each point depends on the values in its neighboring points. As we approach the centre, we will then for each point accumulate greater error, compared to the points closer to the boundary, where we of course know the exact values.

We have previously stated that the presented scheme should be of order $p = 2$, and this has also been tested numerically. From the log-log plot in figure 4, where the slope estimates our order, we see that we get an estimate of $p = 2.00$. From this plot, one might also observe that our previously derived error bound seems rather loose, in the sense that there is a significant distance between our error plot and the bound. A possible reason for this might be that our test function contains trigonometric terms of both x and y, so we have five terms that we need to maximize in our error bound.

We therefore also test our scheme on the polynomial function $u(x, y) = x^4 + y^3$. The point of testing this simpler function is to get a simpler error bound with only one term to maximize, so that we hopefully get a closer error bound. For this particular function, with $a = 1$, we have the bound from equation 1 as

$$\|\mathbf{e}_h\|_\infty \leq \frac{1}{2}h^2$$

The relevant plots are in figure 5, where we can see that the error is noticeably closer to the bound compared to in figure 4. We again get an estimated convergence rate of $p = 2.00$, supporting our theoretical results.

1.4 d) - Bad directions and complications

We now want to use an irriational r, and this creates a few challenges for our previous grid. Firstly, we change k from $2h$ to $|r|h$, so that the directional derivative will follow the diagonal gridlines. Secondly, we will not have nodes along the upper boundary, at $y = 2$, as r is irrational. Since we have $y_n = |r|hn$, we would need $y_N = |r|\frac{N}{M} = 2$. This would lead to $|r| = \frac{2M}{N}$, but if r is irrational, this is not possible.

This can be solved by "fattening the boundary", which we do by making the last horizontal grid line go less than k past our original boundary at $y = 2$. To approximate the value at the points outside of Ω, we take the orthogonal projection onto Ω and see that a node (x_i, y_j) will simply be projected to $(x_i, 2)$. Then we just apply the same scheme as earlier.

Furthermore, if r has a negative sign, the situation changes even more. We will now have a $\mathbf{d}_2$ that points downward, and we will again miss the boundary $y = 2$. Our original stencil will now be "mirrored", and is shown in figure 3. In order to account for this, we also need to flip our scheme, which yields $A = \text{tridiag}\{C, B, C^\top\}$, and we have to update $\mathbf{b}$ accordingly.

In addition, we have tested this implementation for the same polynomial as earlier, $u(x, y) = x^4 + y^3$, as we can then compare how the irrational r affects the solution compared to our previous one. As the function increases toward the upper boundary $y = 2$, we expect to see a larger error here. The numerical solution and error is presented in figure 6. We see that we get a much larger error towards the upper boundary now than previously. We also note that we approximate the order to $p = 1.77$. The reduction of the convergence rate is expected from fattening the boundary. However, this is not completely in accordance with the results we have been presented with in lectures, namely that the rate of convergence should be linear near the fattened boundary. One possible explanation for this might be that this linearity is more easily observed for some choices of step lengths than others.

Another interesting test function is the function $u(x, y) = \sqrt{5 - x^2 - y^2}$. This function is defined within our domain, but we note that for our uppermost right corner, i.e. $(0, 2)$, the argument in the root is zero, and increasing x or y any more from this point would lead to taking the root of a negative number. In the case where $r = 2$, this should pose little to no problems, as we do not require any points outside of our domain. However, when we now have fattened the boundary, we see the need for projecting onto our boundary even more clearly, as this prevents invalid arguments in our test function, compared to using the function itself as the extension outside the domain. From the figure 7 we also see that the error increases the most towards this critical point, and this might be caused by problems with continuity or differentiability.

2 Problem 2

We now want to consider an irregular grid enclosed by the positive x-axis, the positive y-axis and the parabola $y = 1 - x^2$. We want to utilize Dirichlet boundary conditions and we set $\kappa = I$, meaning that we consider an isotropic heat distribution in the material:

$$-\partial_x^2 u - \partial_y^2 u = f$$

We want to approach this problem with two different stategies: by "fattening the boundary" and by "modifying our discretization". In both cases we will need to use the five-point stencil (with some alterations), as discussed in class:

$$-\mathcal{L}_h U_P = 4U_P - U_N - U_E - U_W - U_S$$

2.1 Fattening the boundary

We will start with fattening the boundary. The idea is to extend the boundary by $\sqrt{2}h$, and the points that lie between our original boundary and this "fattened" boundary will be our boundary points in the scheme. We will then approximate the values in the nodes outside of our domain by using the value at the orthogonal projection to our domain Ω. We can then apply our earlier scheme for all the inner nodes. Since the curve $p(x) = 1 - x^2$ is smooth and injective, we were able to find an analytic expression for the fattened boundary. This makes it easier to distinguish between points that would lie within our domain and points that would lie within our fattened boundary. From this we deduced that

$$|p'(x)| = \frac{c}{a} = \frac{a}{b} \implies b = \frac{a}{|p'(x)|}$$

From Pythagoras' theorem we have that

$$a = \frac{\sqrt{2}h}{\sqrt{1 + \frac{1}{p'(x)^2}}}$$

And the function describing the fattened boundary is then

$$q(x) = p(x - a) + b = \frac{-4x^4 - 8h^2x^2 - h(-4\sqrt{2}x^2 - \sqrt{2})\sqrt{4x^2 + 1} + 3x^2 + 1}{4x^2 + 1}$$

An illustration is provided in Figure 8.

2.2 Modifying our discretization

We also want to try a different approach, namely to modify the step length near the boundary. The idea is to create new nodes where our gridlines crosses the boundary. The five-point stencil is then altered according to figure 9. A consequence of this is that the step lengths around the boundary will vary, but these are easily calculated by finding the x- or y-value at the boundary node and subtracting the corresponding value at the inner node. The next step is then to calculate the new coefficients for the central difference. This can be achieved with the formulas from week 6, and has to be done for every node next to the curve. From these formulas, we see that a shorter step length up to the boundary affects the coefficients of the points P, N' and S; and similarly for P, E' and W in the horizontal direction. So we have to update the rows in A corresponding to nodes next to the boundary, and change $\mathbf{b}$ for the same nodes.

To find the modified coefficients, we can evaluate $-\partial_x^2 u$ and $-\partial_y^2 u$ separately.

From the lectures, we know that we get the coefficients

$$\begin{aligned} a_{E'} &= -\frac{2}{h^2}\frac{1}{\eta_x(1 + \eta_x)} \\ a_W &= -\frac{2}{h^2}\frac{1}{1 + \eta_x} \\ a_P &= \frac{2}{h^2}\frac{1}{\eta_x} \end{aligned}$$

for $-\partial_x^2$, while we for $-\partial_y^2$ get the same coefficients, except that η_x is replaced by η_y.

2.3 Results and comparison of the methods

When implementing both methods, we did this in the same order as presented in the report. This meant that after having found a way of constructing the matrix A for fattening the boundary,

it was not too difficult to extend this to the case of modifying the discretization. However, matching up the different coefficients for the modification also provided some tribulations, and we would classify fattening the boundary as the easier method to implement.

When it comes to the performance of the two methods, we have plotted the numerical solutions and error plots for both methods for the test function $u(x, y) = 1 + \cos^2(2\pi x) + \sin^3(3\pi y)$. The plots for fattening the boundary are presented in figure 10, while the plots for modifying the discretization is presented in figure 11.

From the plots we observe that both methods again perform rather well inside our domain, but that the error increases towards the curved part of the boundary. From the log-log plots and the error plot, we also note that the errors of both methods, measured in the inf-norm, are relatively similar in magnitude. This is also true for the numerically estimated convergence rates, which both are close to $p \approx 1$. These results are in better concord with the theory we discussed in 1d) than the results we presented there, which might be because we are dealing with a more complicated boundary.

2.3.1 Timing the methods

We have timed each method by running the cell magic command `%% timeit`. For fattening the boundary we achieve the following result:

[1]:
```
%%timeit -r 5 -n 10
x, y, U = solve_PDE_fat(g, f, M=1000)
```

```
8.99 s ± 141 ms per loop (mean ± std. dev. of 5 runs, 10 loops each)
```

while we got the following result for modifying the discretization:

[2]:
```
%%timeit -r 5 -n 10
x, y, U = solve_PDE_mod(g, f, M=1000)
```

```
8.68 s ± 124 ms per loop (mean ± std. dev. of 5 runs, 10 loops each)
```

The results are very similar, with 'fattening the boundary' being a little slower. For both methods, we have used sparse matrices and methods to solve sparse systems; in addition to using array slicing to alter the systems. The main parts of the algorithms are therefore similar. However, modifying the scheme has to spend some time altering the system matrix A with the modified coefficients, while fattening the boundary spends time finding the projections of the grid points, which could explain the small time difference.

3 Conclusion

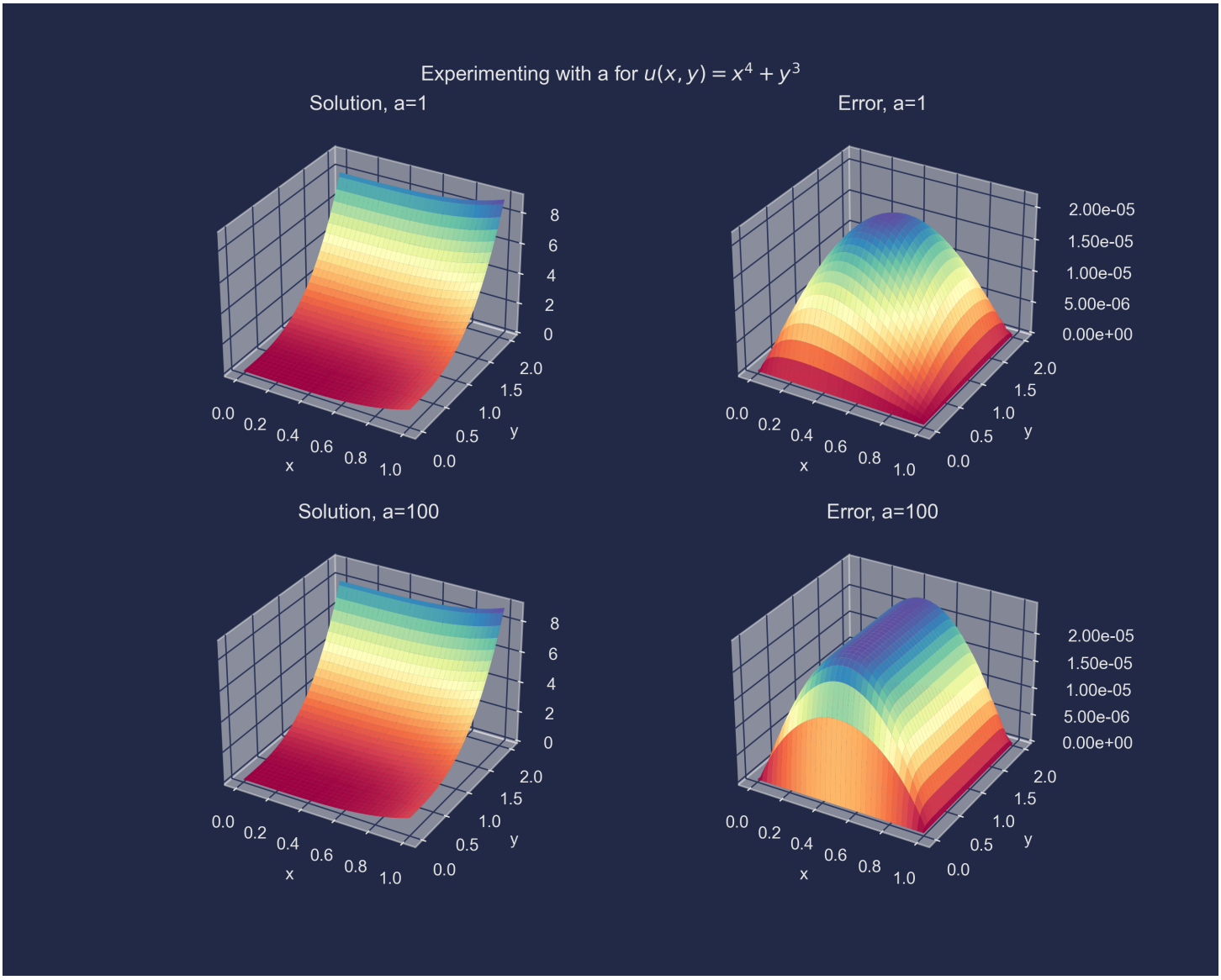
After having implemented different stencils, domains and grids, it is clear that any minor change to the task creates multiple new challenges. In this project we have utilized features that are specific to this exact problem, for instance when it comes to the geometry of our domain. However, having implemented our schemes in Python and tested it, we feel that we have learned valuable lessons that might prove useful in other situations. Overall, we have also seen that the numerics match quite well with the analytical results, as well as with the theory. We have seen that we got order $p = 2$ when we used central differences on a rectangular domain in 1c), and that irregular domains lowered this order to $p \approx 1$ in problem 2.

References

[1] Brynjulf Owren (2017) TMA4212 Numerical solution of partial differential equations with finite difference methods

4 Appendix

Figure 1: 1a: Experimenting with different values for a



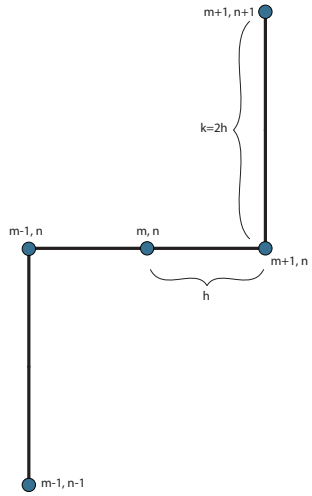


Figure 2: Stencil with $k = 2h$

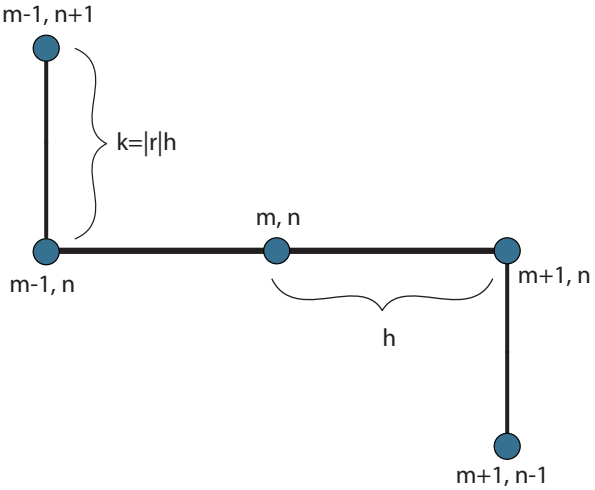


Figure 3: Stencil with $k = |r|h, r < 0$

Figure 4: 1c: Trigonometric test problem

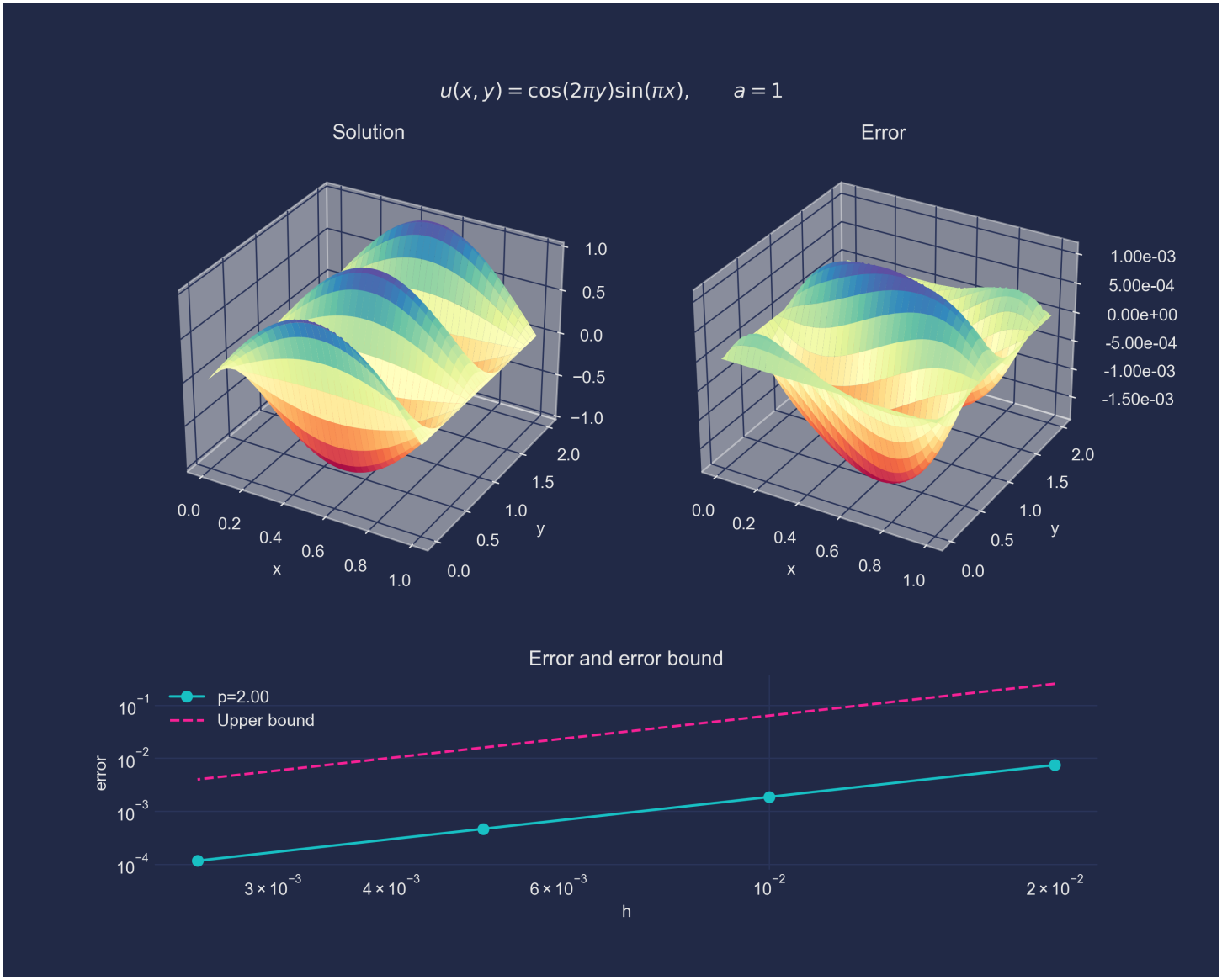


Figure 5: 1c: Polynomial test problem

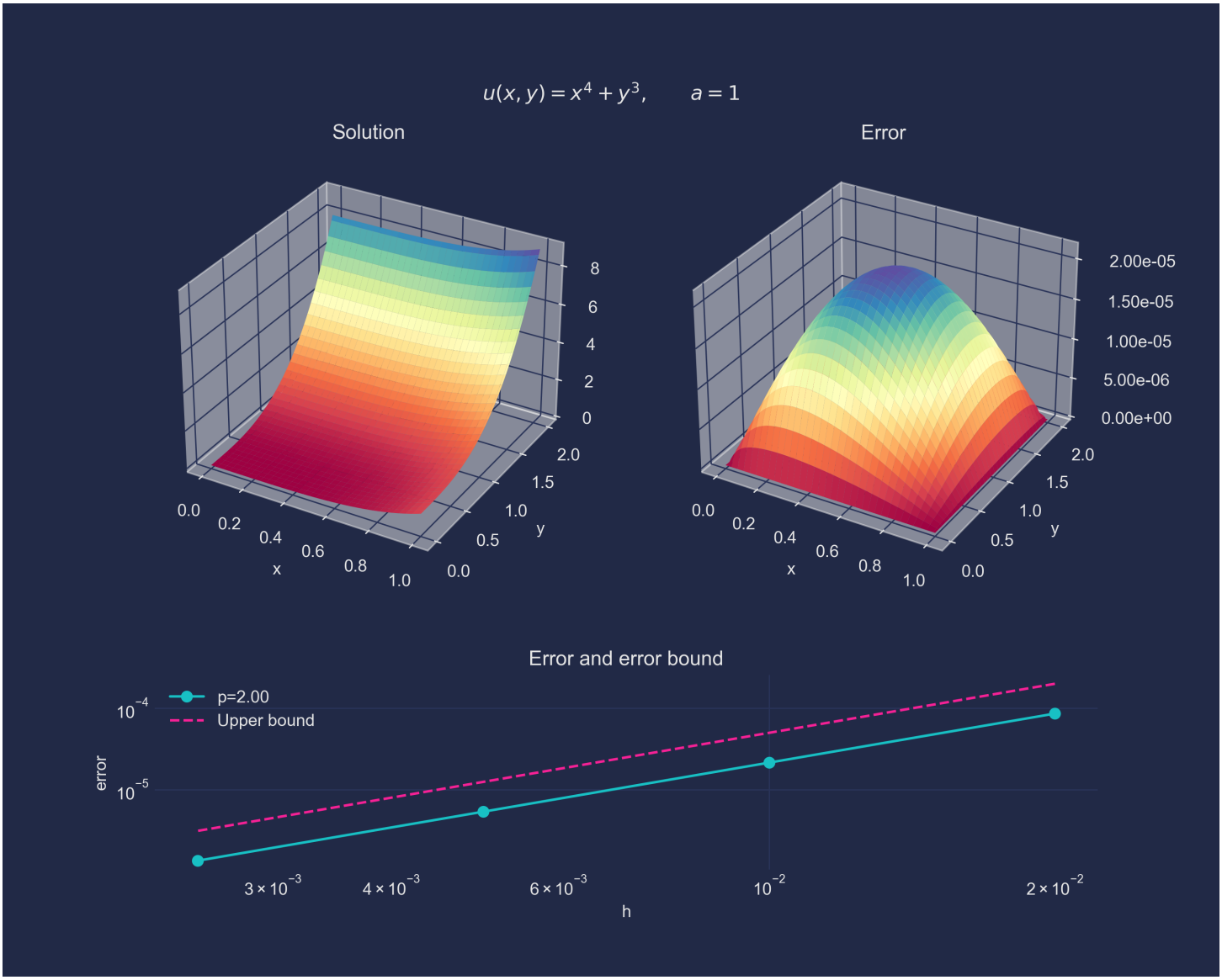


Figure 6: 1d: Polynomial test problem

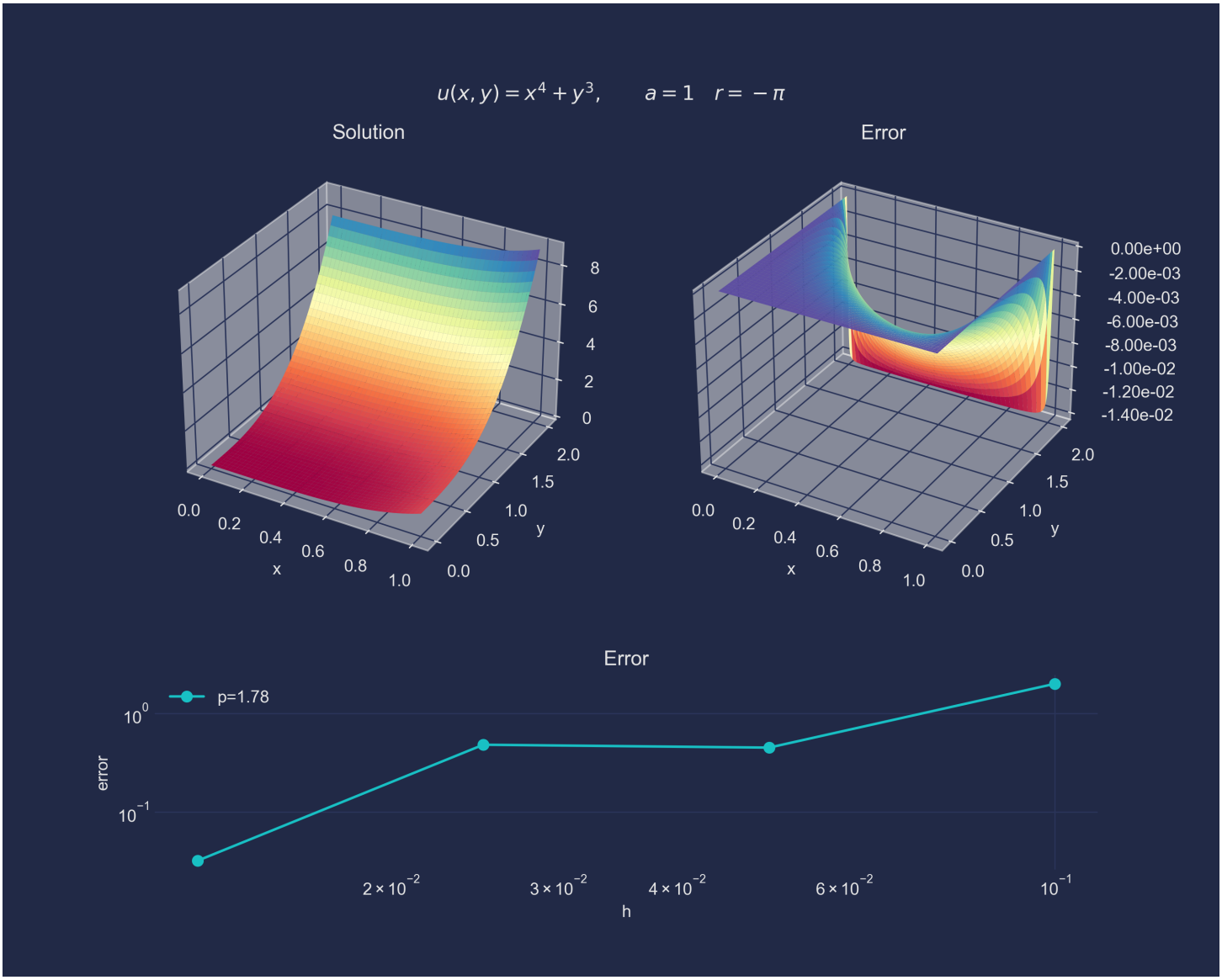


Figure 7: 1d: Test problem that is undefined outside of the grid

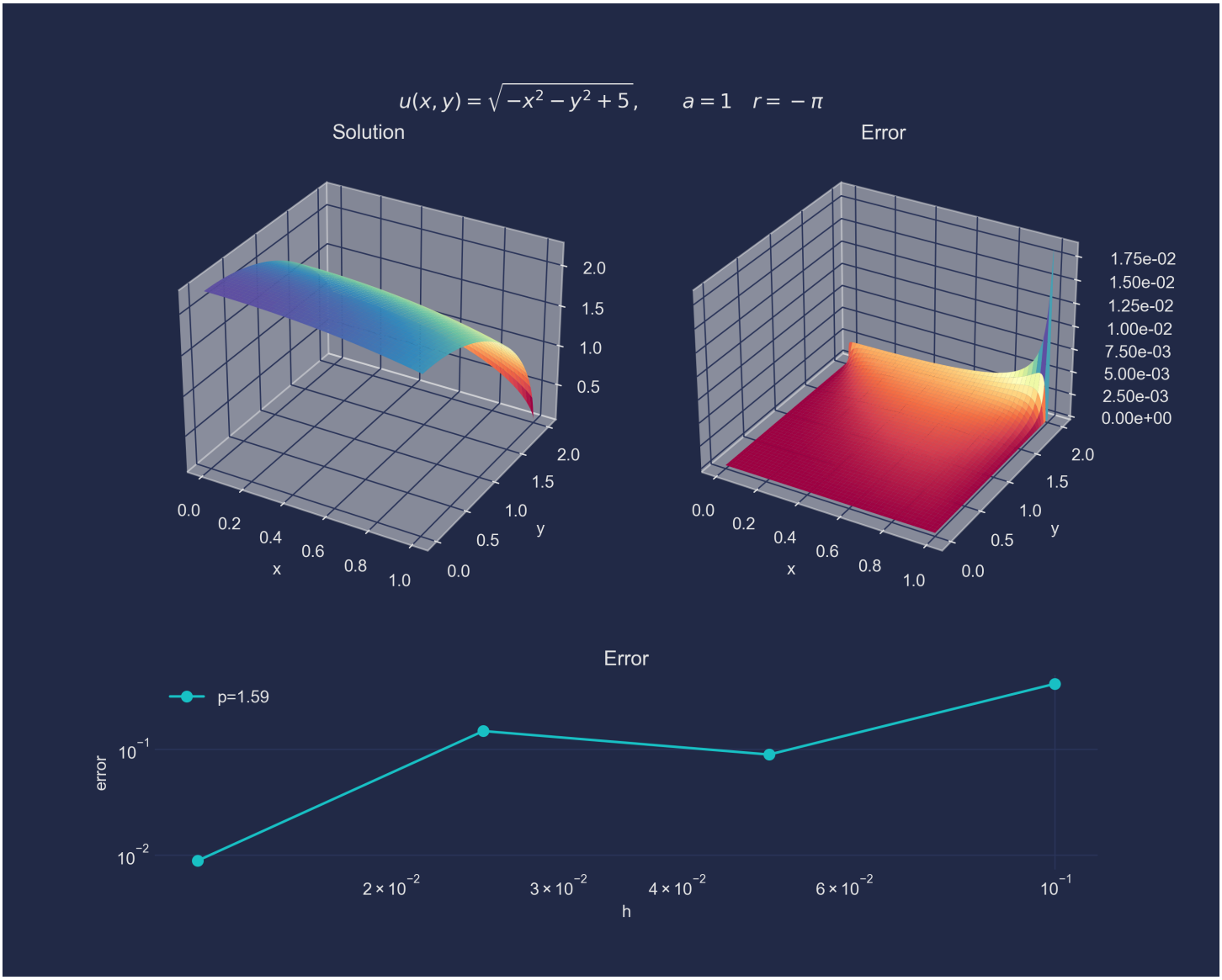


Figure 8: Fatten the boundary

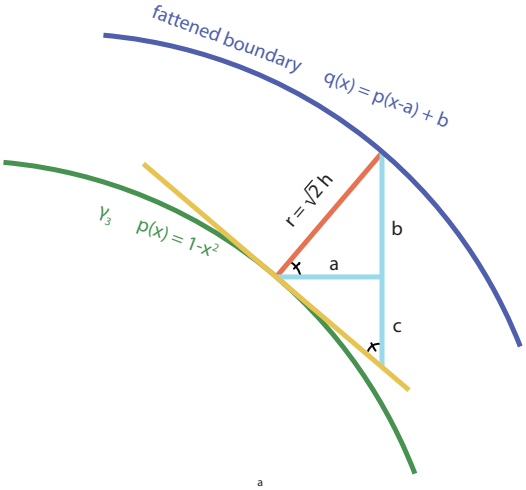


Figure 9: Modified five-point stencil

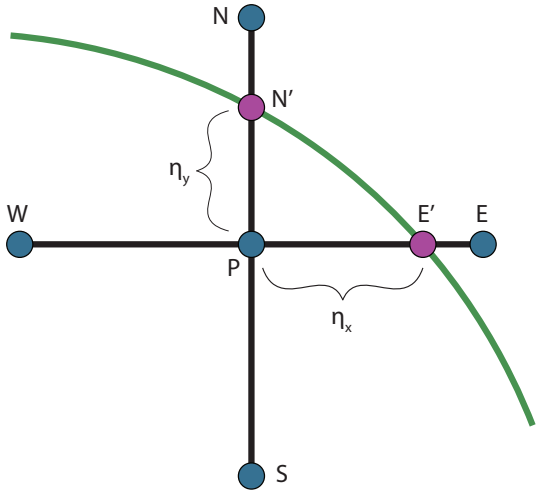


Figure 10: 2.1: Trigonometric test problem by 'fattening the boundary'

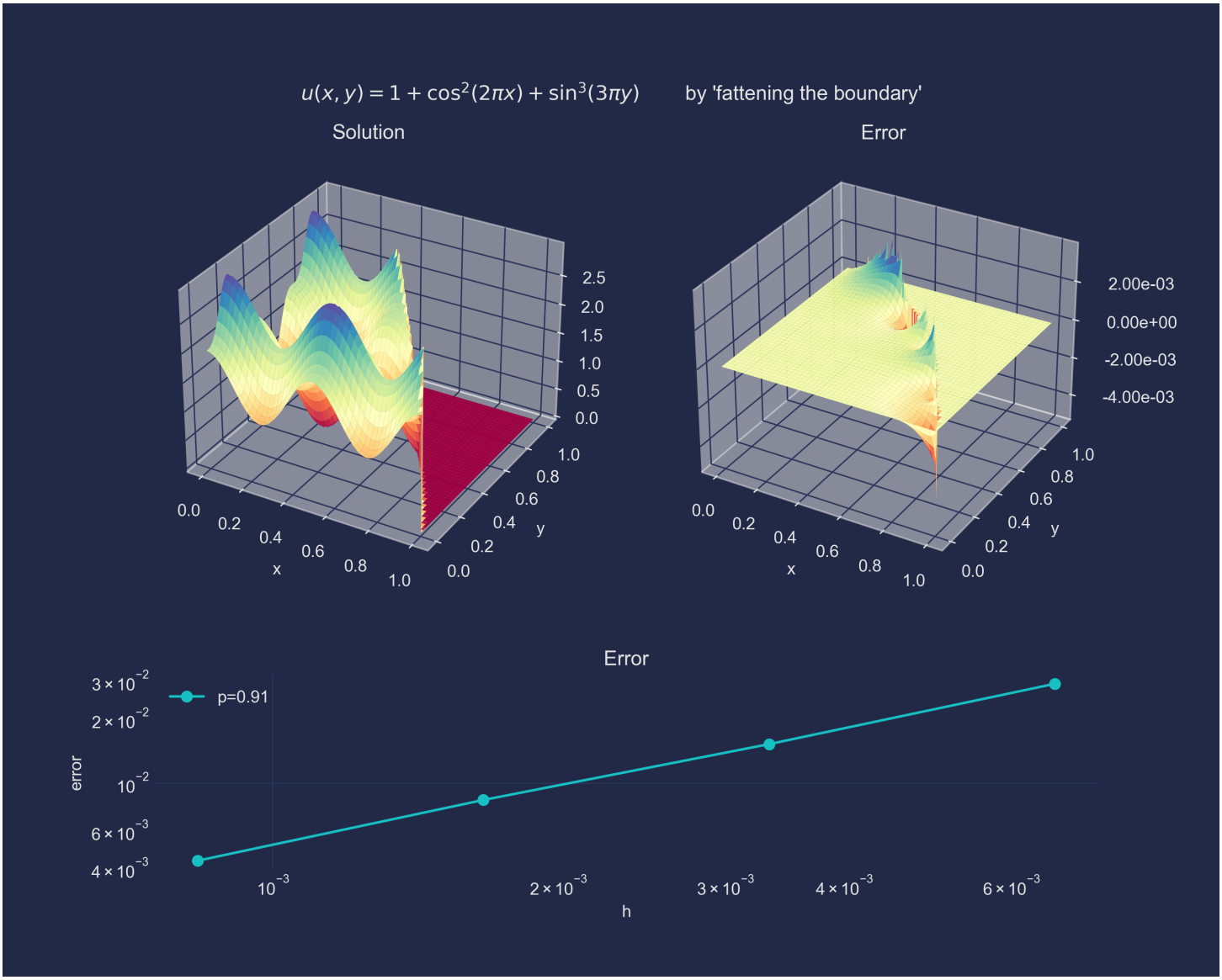


Figure 11: 2.2: Trigonometric test problem by 'modifying the discretization'