TMA4212 - NUMDIFF

Project 1

Authors:
Lars Vatten, Jakob Heide, Celine Olsson

26.02.23

Table of Contents

Introduction

The Poisson equation is used to model the stationary temperature distribution T of a solid Ω. Given a heat conductivity κ and if the solid has internal heat sources f, the conservation of energy and stationarity will give the model

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

We have two directions for the heat flow, $\vec{d_1} = (1, 0)$ and $\vec{d_2} = (1, r)$, where $r \in \mathbb{R}$.

This gives a heat conductivity of the form

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

where $a > 0$ is constant. We then get

$$\nabla \cdot (\kappa \nabla T) = a\partial_x^2 u + (\vec{d_2} \cdot \nabla)^2 u \tag{2}$$

Part 1

In this part we consider the domain $\Omega = [0, 1] \times [0, 2]$, and use Dirichlet boundary conditions $u = g$ on the boundary $\partial\Omega$.

a

Let $r = 2$, and consider the grid $\mathbb{G} = \partial\mathbb{G} \cup \overline{\mathbb{G}}$, where $\partial\mathbb{G}$ and $\overline{\mathbb{G}}$ is the boundary and inner part of $\mathbb{G}$, respectively. Let $h = \frac{1}{M}$ denote the step size in the x-direction and $k = rh$ denote the step size in the y-direction. We want to create a numerical scheme of (1) by replacing the derivatives by second order central difference approximations in the directions $\vec{d_1}$ and $\vec{d_2}$. Discretising the first term of equation (2), we get the approximation

$$a\partial_x^2 u_m^n \approx a\delta_x^2 u_m^n = \frac{a}{h^2}(U_{m+1}^n - 2U_m^n + U_{m-1}^n) = O(h^2)$$

The second term of equation (2) can be written as

$$(\vec{d_2} \cdot \nabla)^2 u = \partial_x^2 u + 2r\partial_x\partial_y u + r^2\partial_y^2 u. \tag{3}$$

However, we do not want to discretise using mixed derivatives, since this might yield an unstable scheme. Instead, we want to use a second order central difference approximation along the direction of $\vec{d_2}$. Let $\delta_{\vec{d_2}}^2$ denote the second order central difference approximation in the direction $\vec{d_2}$. We want a scheme on the form

$$(\vec{d_2} \cdot \nabla)^2 u_m^n \approx \delta_{\vec{d_2}}^2 u_m^n = aU_{m+1}^{n+1} + bU_m^n + cU_{m-1}^{n-1}.$$

In order to solve for the unknown coefficients a, b and c, we impose the condition that the local truncation error $\tau_m^n = (\vec{d_2} \cdot \nabla)^2 u_m^n - \delta_{\vec{d_2}}^2 u_m^n$ should be $O(h^2 + k^2)$.

First we Taylor expand U_{m+1}^{n+1} and U_{m-1}^{n-1}:

$$\begin{aligned} U_{m+1}^{n+1} &= u_m^n + hu_{x,m}^n + ku_{y,m}^n + \frac{h^2}{2}u_{xx,m}^n + hku_{xy,m}^n + \frac{k^2}{2}u_{yy,m}^n \\ &\quad + \frac{h^3}{6}u_{xxx} + \frac{h^2k}{2}u_{xxy} + \frac{hk^2}{u_{xyy}} + \frac{k^3}{6}u_{yyy} + O(h^4 + k^4) \\ U_{m-1}^{n-1} &= u_m^n - hu_{x,m}^n - ku_{y,m}^n + \frac{h^2}{2}u_{xx,m}^n + hku_{xy,m}^n + \frac{k^2}{2}u_{yy,m}^n \\ &\quad - \frac{h^3}{6}u_{xxx} - \frac{h^2k}{2}u_{xxy} - \frac{hk^2}{u_{xyy}} - \frac{k^3}{6}u_{yyy} + O(h^3 + k^3) \end{aligned}$$

Inserting this into the discretisation formula, and subtracting from the expression in (3), we get the following expression for the local truncation error:

$$\begin{aligned}\tau_m^n =&(-a-b-c)u_m^n + h(-a+c)u_{x,m}^n + k(-a+c)u_{y,m}^n + \frac{h^2}{2}(1-a-c)u_{xx,m}^n \\ &+ \frac{hk}{2}(2r-a-c)u_{xy,m} + \frac{k^2}{2}(r^2-a-c)u_{yy,m}^n + \frac{h^3}{6}(-a+c)u_{xxx,m}^n + \frac{h^2k}{2}(-a+c)u_{xxy,m}^n \\ &+ \frac{hk^2}{2}(-a+c)u_{xyy,m}^n + \frac{k^3}{6}(-a+c)u_{yyy,m}^n + O(h^4+k^4)\end{aligned}$$

Imposing the condition $\tau_m^n \overset{!}{=} O(h^2+k^2)$, we get a set of equations to solve for the coefficients of our discretisation formula, where all the terms that are not $O(h^2+k^2)$ has to cancel to zero. We solve the equations and get the following coefficients:

$$a = c = \frac{1}{h^2} \qquad b = -\frac{2}{h^2}$$

We see that the equations we solved for makes sure that the $O(h^3+k^3)$ terms from the expression above cancel. Since all coefficients are proportional to $\frac{1}{h^2}$, we are left with $\tau_m^n = O(h^2+k^2)$, which is what we wanted to obtain. The discretisation formula for the second order derivative in the direction of $\vec{d_2}$ becomes

$$(\vec{d_2}\cdot\nabla)^2 u = \frac{1}{h^2}U_{m-1}^{n-1} - \frac{2}{h^2}U_m^n + \frac{1}{h^2}U_{m+1}^{n+1}.$$

Inserting our approximation formulas for the derivatives into (2) yields the numerical scheme

$$\begin{aligned}\nabla\cdot(\kappa\nabla T) \approx -L_hU_p &= \frac{a}{h^2}U_{m-1}^n - \frac{2(a+1)}{h^2}U_m^n + \frac{a}{h^2}U_{m+1}^n + \frac{1}{h^2}U_{m-1}^{n-1} + \frac{1}{h^2}U_{m+1}^{n+1} \\ &= A\vec{U^n} + B\vec{U^{n-1}} + C\vec{U^{n+1}} = DU\end{aligned} \tag{4}$$

where A, B and C are tridiagonal matrices and D is a block-tridiagonal matrix on the form:

$$A = \text{tridiag}\left(\frac{a}{h^2}, -\frac{2(a+1)}{h^2}, \frac{a}{h^2}\right), \quad B = \text{tridiag}\left(\frac{1}{h^2}, 0, 0\right), \quad C = \text{tridiag}\left(0, 0, \frac{1}{h^2}\right)$$
$$D = \text{tridiag}(B, A, C)$$

This scheme gives the stencil illustrated in Figure 1. Where (x_m, y_n) is the central point (P), (x_{m-1}, y_n) is the point west of P (W), (x_{m+1}, y_n) is the point east of P (E), (x_{m-1}, y_{n-1}) is the point south-west of P (SW), and finally (x_{m+1}, y_{n+1}) is the point north-east of P (NE).

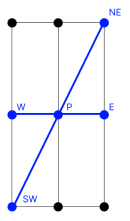


Figure 1: Illustration of the stencil

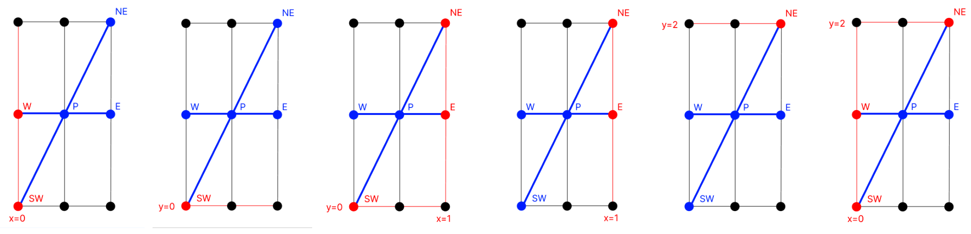
The right hand side of this scheme is $-f$ evaluated in every inner point of the grid. Since the scheme only solves for the inner points, we need to add the contribution from the boundary points for the inner points in the grid for which the stencil includes points on the boundary.

In the case of a rectangular domain, the stencil in Figure 1 gives rise to six different cases at the boundary - bottom left corner/left edge, bottom edge, bottom right corner, top right corner/right edge, top edge and top left corner. The different cases are illustrated in Figure 2. If a point P is subject to one of these six cases, the boundary points are evaluated by the boundary function g, and added on the right hand side, weighted by the corresponding coefficient given by the scheme.

If we set $b = -f + g$ to be the vector of f evaluated in each inner point summed with the contribution of the boundary g for inner point, we get the simplified system

$$DU = b, \tag{5}$$

which we can solve numerically with known methods, e.g. `NumPy.linalg`'s `solve()`-function.



(a) (b) (c) (d) (e) (f)

Figure 2: (a) Bottom left corner, left edge (b) Bottom edge (c) Bottom right corner (d) Top right corner, right edge (e) Top edge (f) Top left corner

b

We now consider some important properties for performance of the scheme implemented in part 1a) - namely monotonicity, L^∞-stability and convergence.

Monotonicity (positive coefficients)
A scheme on the form

$$-L_h U_p = \alpha_{pp} U_p - \sum_{Q \neq P} \alpha_{PQ} U_Q = F_P$$

is said to have positive coefficients if $\alpha_{pp} \geq 0$ and $\alpha_{pp} \geq \sum_{Q \neq P} \alpha_{PQ}$. We can write our scheme (4) as

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

With $a > 0$ we see that our scheme has positive coefficients. Our scheme is also boundary connected, and thus monotone, which is a property we will use in the following discussion.

L^∞-stability
We want to show that any solution of (4) is stable in L^∞ w.r.t. some RHS f. Let $\phi(x) = \frac{1}{2}x(1-x)$. Note that by applying our discretised differential operator $-L_h$ we get $-L_h \phi_P = 1 + a \geq 1$. Assume that V_P is a solution of (4) with $V_P = 0 \; \forall P \in \partial \mathbb{G}$. Let $W_P := V_P - \phi_P \|\vec{f_P}\|_\infty$ and consider

$$\begin{aligned} -L_h W_P &= -L_h V_P - (-L_h \phi_P)\|\vec{f_P}\|_\infty \\ &= f_p - (1+a)\|\vec{f_P}\|_\infty \leq 0 \end{aligned}$$

By the result of monotonicity we can now apply the discrete maximum principle:

$$\begin{aligned} -L_h W_P \leq 0 \xRightarrow{\text{(DMP)}} \max_{P \in \mathbb{G}} W_P &\leq \max_{P \in \partial \mathbb{G}} \{W_P, 0\} \\ \Rightarrow V_P - \phi_P \|\vec{f_P}\|_\infty &\leq 0 \\ \Rightarrow V_P &\leq \phi_P \|\vec{f_P}\|_\infty \end{aligned}$$

The transformation $(V, f) \mapsto (-V, -f)$ gives the similar result $-V_p \leq \phi_P \| - \vec{f_P}\|_\infty$. Taking the max and absolute value on both sides, we get the stability result

$$\max |V_P| \leq \max \phi_P \|\vec{f_P}\|_\infty = \frac{1}{8} \|\vec{f_P}\|_\infty$$

where the last equality stems from the fact that $\max_{x \in [0,1]} \phi(x) = \frac{1}{8}$.

Convergence
Consider the error equation $e_P = u_P - U_P$. Applying our operator $-L_h$ we find

$$-L_h e_P = -L_h u_P - (-L_h U_P) = -L u_P - \tau_P - f_p = -\tau_P$$

Note that the error e_P satisfies $e_P = 0$ on $\partial\mathbb{G}$. We can therefore apply the stability result found before:

$$\|e_P\|_\infty \leq \frac{1}{8}\|\tau_P\|_\infty$$

We derive an error bound by deriving the expression for the local truncation error in a point (x_m, y_n). The local truncation error τ_m^n is given by

$$\tau_m^n = Lu_m^n - L_hU_m^n = (a+1){u_{xx}}_m^n + 2r{u_{xy}}_m^n + r^2{u_{yy}}_m^n - (\delta_x^2 u_m^n + \delta_{d_2}^2 u_m^n)$$

If we Taylor expand the terms in the central difference formulas in L_h so that the truncation error is expressed as a linear combination of evaluations of u and its derivatives in the point (x_m, y_n), we obtain the following expression for the error after carrying out some algebra:

$$\tau_m^n = -\frac{(a+1)h^2}{12}{u_{xxxx}}_m^n - \frac{rh^2}{3}{u_{xxxy}}_m^n - \frac{r^2h^2}{2}{u_{xxyy}}_m^n - \frac{r^3h^2}{3}{u_{xyyy}}_m^n - \frac{r^4h^2}{12}{u_{yyyy}}_m^n + O(h^4)$$

Hence the error bound e_P becomes:

$$\|e_P\|_\infty \leq \frac{h^2}{8}\|\frac{(a+1)}{12}u_{xxxx,P} + \frac{r}{3}u_{xxxy,P} + \frac{r^2}{2}u_{xxyy,P} + \frac{r}{3}u_{xyyy,P} + \frac{r^4}{12}u_{yyyy,P}\|_\infty \tag{6}$$

c

We define some functions to test out the implementation we made in part 1a).

$$u_1(x,y) = x^3y^3, \quad f_1(x,y) = 6(a+1)xy^3 + 18rx^2y^2 + 6r^2x^3y \tag{I}$$
$$u_2(x,y) = \cos x \sin y, \quad f_2(x,y) = -(a+1)\cos x \sin y - 2r \sin x \cos y - r^2 \cos x \sin y \tag{II}$$

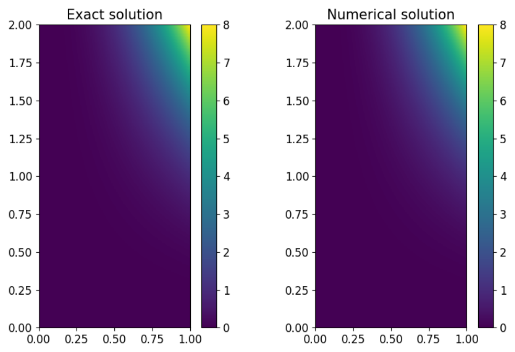


Figure 3: Solutions using (I)

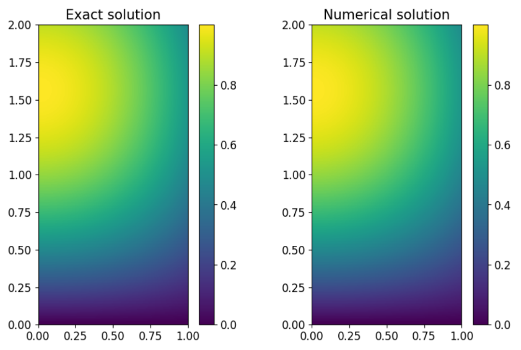


Figure 4: Solutions using (II)

We can see from Figure 3 and 4 that the numerical solution we get from our scheme is indistinguishable from the exact solution. But how close are we really?

Using the same test functions we now look at the error. We use equation (6) to calculate an error bound and find an upper limit. An error bound can be found simply by finding the maximum of each single term in (6). This will not give us the tightest bound, but it is easier to compute.

Since $x \in [0,1]$ and $y \in [0,2]$, the maximum value x and y can take is 1 and 2.

Using this in case (I) we get that

$$\max_{x,y} u_{xxxx} = 0, \quad \max_{x,y} u_{xxxy} = 48, \quad \max_{x,y} u_{xxyy} = 48, \quad \max_{x,y} u_{xyyy} = 12, \quad \max_{x,y} u_{yyyy} = 0$$

The theoretical upper bound for the error of (I) (when using $r = 2$) will then be

$$||e_p||_\infty \leq \frac{h^2}{8}\left(\frac{a+1}{12}\cdot 0 + \frac{r}{3}48 + \frac{r^2}{2}48 + \frac{r^3}{3}12 + \frac{r^4}{12}\cdot 0\right) = 20h^2$$

Doing the same calculation for (II) and considering that $\cos x$ and $\sin x$ have a maximum and minimum value of 1 and -1, we get

$$\max_{x,y} u_{xxxx} = \max_{x,y} u_{xxxy} = \max_{x,y} u_{xxyy} = \max_{x,y} u_{xyyy} = \max_{x,y} u_{yyyy} = 1$$

and the theoretical upper bound for the error of (II) (using that $r = 2$) will then be

$$||e_p||_\infty \leq \frac{h^2}{8}\left(\frac{a+1}{12}\cdot 1 + \frac{r}{3}\cdot 1 + \frac{r^2}{2}\cdot 1 + \frac{r^3}{3}\cdot 1 + \frac{r^4}{12}\cdot 1\right) = \frac{41}{48}h^2$$

Below is a loglog-plot of the error and the upper bound for both cases.

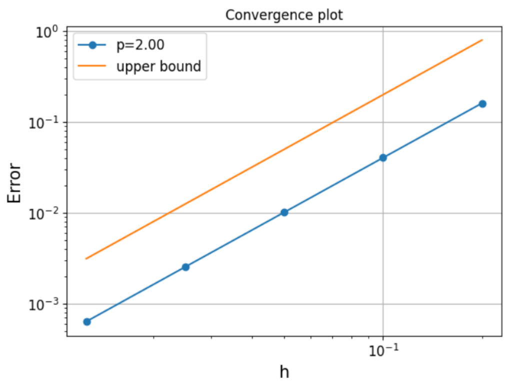


Figure 5: Error using (I)

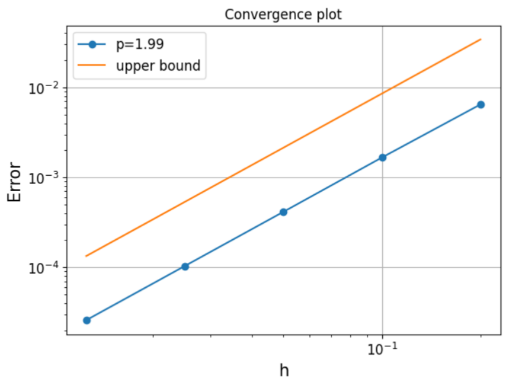


Figure 6: Error using (II)

We see that both methods give a convergence of approximately order 2, which is expected, as the error of our method is $O(h^2)$. We also see that the theoretical upper bounds for (I) and (II) hold for both cases.

d

Now we set r to be $r \neq 2$ and irrational, and introduce a new grid with step size $h = 1/M$ in x direction and $k = |r|h$ in y direction. Using the same approach as in part 1a) we get the same expression for $(\vec{d_2}\cdot\nabla)^2 u$, which gives us the same discretised scheme (4).

Since the step size in the y direction now is irrational, the total length of the grid will also be irrational. This means that the grid will now miss the upper boundary of $y = 2$, which is a rational number. To solve this problem we decided to fatten the boundary of the grid, which we achieve by extending the boundary slightly outside of the domain. This allows us to keep a regular grid, which is easier to work with. We handle the boundary conditions by applying the boundary conditions from $y = 2$ to the new boundary. More technically, we impose that the value in a point in the grid which is outside of the actual boundary should be equal to the value of the boundary function evaluated in the given point projected down on the boundary. Luckily, since we are dealing with a rectangular domain, the projected points does not depend on r, so the solver is almost identical to before.

Using (II) from part 1c) we test the new implementation of the solver. Below is the solutions (7) and error (8) when using $r = \sqrt{2}$.

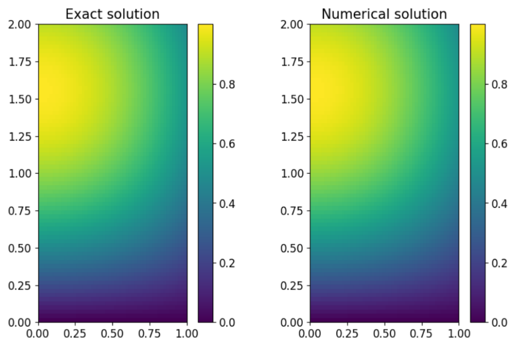


Figure 7: Solution using $r = \sqrt{2}$

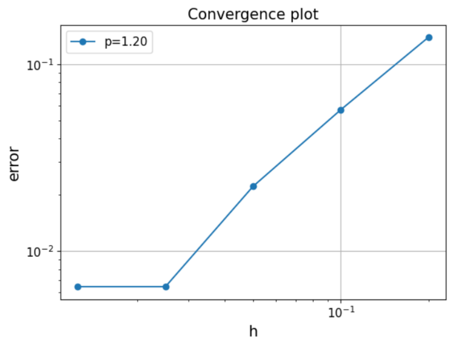


Figure 8: Error when using $r = \sqrt{2}$

Part 2

In this part we consider the domain Ω to be enclosed by the parabola $y = 1-x^2$, the x-axis and the y-axis, and assume that step sizes in x and y direction are equal ($h = k = 1/M$). Our boundary now consist of the three different curves

$$\gamma_1 = [0,1] \times \{0\}, \quad \gamma_2 = \{0\} \times [0,1], \quad \gamma_3 = \{(x, 1-x^2) : x \in [0,1]\}$$

We want to solve numerically the Dirichlet boundary value problem for (1), only now $\kappa = I$ and $\nabla \cdot (\kappa \nabla T) = \Delta T$. This gives us the new problem

$$\begin{aligned} -\Delta T &= f \quad \text{in } \Omega \\ -(\partial_x^2 u + \partial_y^2 u) &= f \end{aligned} \tag{7}$$

We discretise using second order central difference approximation, and get the 5-point scheme

$$-\Delta T \approx -L_h U_p = \frac{1}{h^2}(-U_{i+1,j} - U_{i-1,j} - U_{i,j+1} - U_{i,j-1} + 4U_{i,j}) = f_p \tag{8}$$

Giving us the stencil in Figure 9.

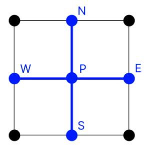


Figure 9: Illustration of the stencil

(i)

We consider first the method of modifying the finite difference scheme for points that are close to the boundary points. Note that we only have to modify the scheme for points close to γ_3. In these points we have to modify the coefficients in 8. We use the method of undetermined coefficients in both the x and y-direction to find the equations in the points close to the boundary. Starting with the y-direction, we get

$$\partial_y^2 u - \tau_{y,P} = a_S U_S + a_P U_P + a_{N'} U_{N'}$$

Similarly to what we did in part part 1a, by Taylor expanding U_S and $U_{N'}$ we get a system of equations for the coefficients a_S, a_P and $a_{N'}$, which we solve to find

$$a_{N'} = \frac{2}{h^2}\frac{1}{\eta_N(\eta_N+1)} \qquad a_P = -\frac{2}{h^2}\frac{1}{\eta_N} \qquad a_S = \frac{2}{h^2}\frac{1}{1+\eta_N}$$

In a similar way we find the coefficients for the discretisation in the x-direction to be

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

We implement this scheme by building the matrix-vector system in a row-by-row fashion, and modifying the scheme in the points close to the boundary. Since the grid is no longer regular the matrix on the left hand side is also no longer block-tridiagonal. Using the test function $u_2(x, y) = \cos x \sin y$, we solve the system for the inner points with $h = 0.01$ and compare the solution with the exact solution in figure 10.

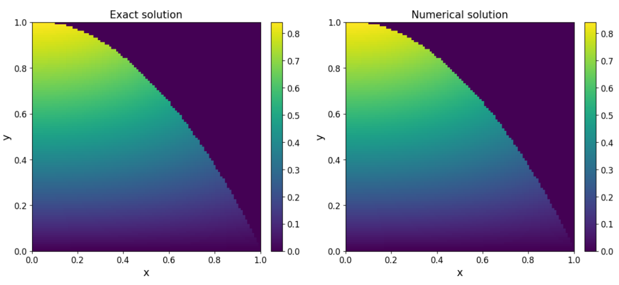


Figure 10: Solutions when using the modified boundary

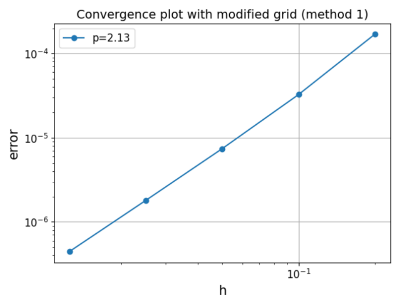


Figure 11: Error using the modified boundary

We see that the scheme is able to reproduce the exact solution in a good way. The convergence rate for this method is shown in 11.

From the lectures we expect the local truncation error $\tau_P = (1 - \eta_N)O(h) + (1 - \eta_E)O(h) + O(h^2)$. The convergence rate in 11 shows that the error behaves as expected, since $O(h) \in O(h^{2.13})$ (for small enough h).

(ii)

We now move on to the method of fattening the boundary. This is done in a similar way to part 1d), but here we calculate the projection from the points in the boundary extension onto γ_3 and use the function values in these points as boundary conditions. The results of this method using $h = 0.01$ are shown in 12.

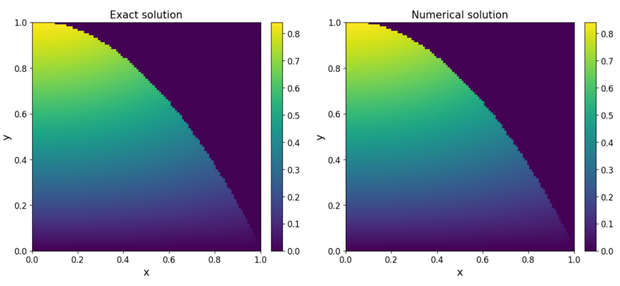


Figure 12: Solutions when fattening the boundary

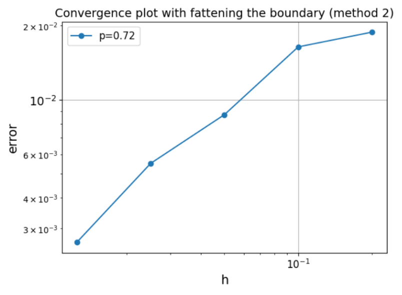


Figure 13: Error using the fattened boundary

We expect the error to be of order $O(h)$ in this method. In 13 the convergence of the method is shown. We see that the convergence rate behaves as expected with order $p = 0.72$.

We also tested the implementation on a different function and gained very similar results, see Appendix 1 and Appendix 2.

The method of fattening the boundary turned out to be much simpler to implement than modifying the grid. We were able to handle boundary conditions of fattening the boundary in two lines (`scheme` function, line 29-30) with the use of a helper function to calculate the projection onto γ_3. The second method of modifying the grid near the boundary took a grand total of 72 lines (`scheme` function, line 36-108)! The solution of the second method is most definitely not optimal, but it is worth noting the difference in effort.

A1: Fattening the boundary

$$u_3(x,y) = \sin 10xy, \qquad f_3(x,y) = \sin 10xy \tag{III}$$

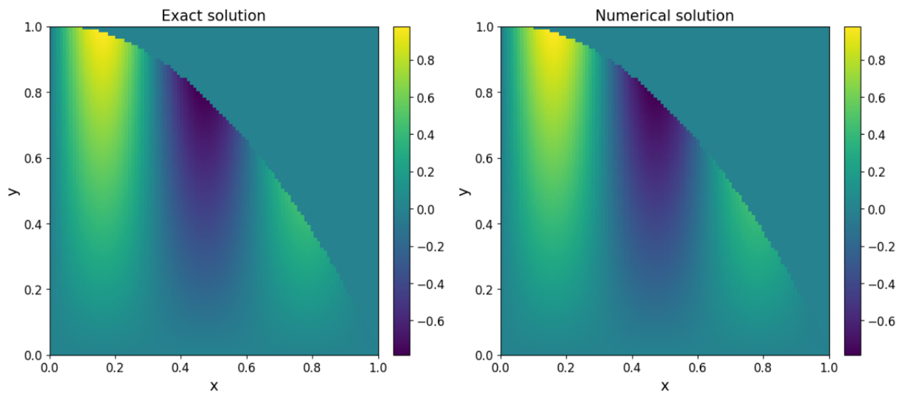


Figure 14: Solutions when fattening the boundary

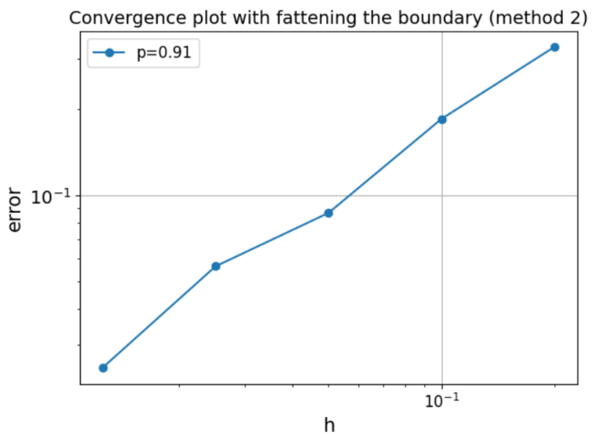


Figure 15: Error when fattening the boundary

A2: Modifying the grid

$$u_3(x,y) = \sin 10xy, \qquad f_3(x,y) = \sin 10xy \tag{III}$$

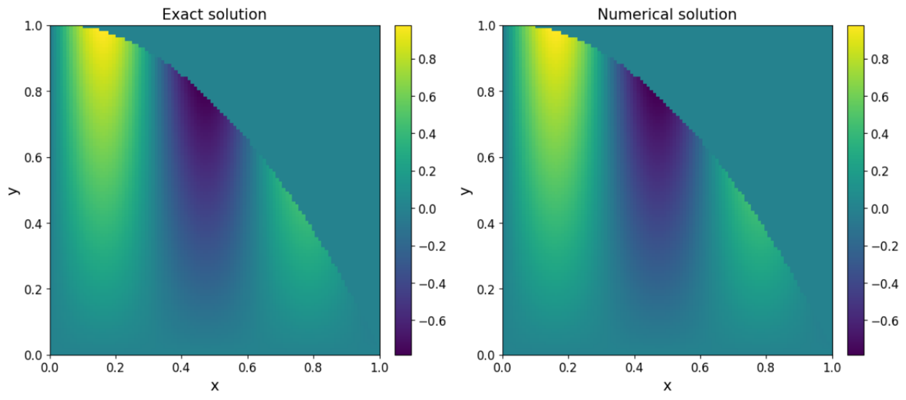


Figure 16: Solutions when modifying the grid

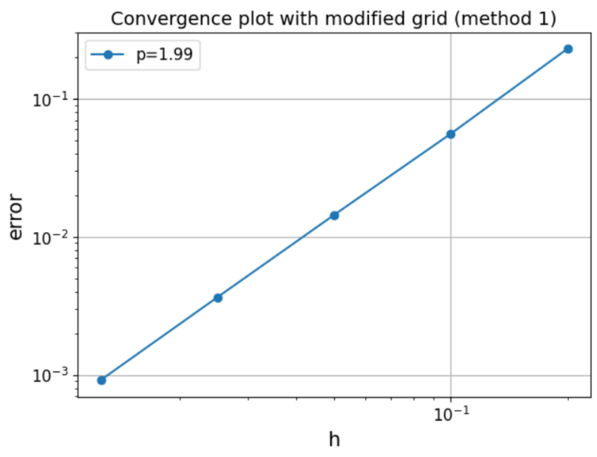


Figure 17: Error when modifying the grid