NTNU
Kunnskap for en bedre verden

Department of Mathematical Sciences

TMA4212 - Numerical solution of differential equations by difference methods

Course Project

Authors:

Marius Lindegaard, Frederick Nilsen, Petter Røe Sundhaug,
Bendik Skundberg Waade

April, 2021

Contents

List of Figures

List of Tables

1 Part 1

1.1 Boundary value problems: Poisson equation

In this part, we consider the one-dimensional Poisson equation, given by

$$u_{xx} = f(x), \quad x \in [0,1] \tag{1}$$

Choosing $f(x)$ and boundary conditions so that we may determine an analytical solution, we can analyze convergence of our constructed finite difference schemes. We will also consider an ill-posed problem with a pure Neumann boundary condition. The segment starts off examining (1) with

$$f(x) = \cos(2\pi x) + x \tag{2}$$

In addition, we consider the specific case with pure Dirichlet conditions and the manufactured solution

$$u(x) = \exp -\frac{1}{\epsilon}\left(x - \frac{1}{2}\right)^2, \tag{3}$$

and preform both uniform and adaptive mesh refinement techniques on the problem to optimize accuracy.

1.1.1 Mathematical formulation

1.1.1.1 Analytical solutions of well-posed BVPs

We first determine the analytical solution for the Poisson equation with $f(x)$ defined by (2), i.e.

$$u_{xx} = \cos(2\pi x) + x,$$

We integrate twice to get

$$u_x = \frac{1}{2\pi}\sin(2\pi x) + \frac{1}{2}x^2 + c_1, \qquad c_1 \in \mathbb{R}$$
$$u = \frac{-1}{4\pi^2}\cos(2\pi x) + \frac{1}{6}x^3 + c_1 x + c_2, \qquad c_1, c_2 \in \mathbb{R}$$

We now determine the values for c_1 and c_2 based on the following boundary conditions.

a) $u(0) = 0 = u_x(1)$, Dirichlet on left and Neumann on right

b) $u(0) = 1 = u(1)$, Dirichlet on both sides

c) $u_x(0) = 0, u_x(1) = \frac{1}{2}$, Neumann on both sides

For the first set of boundary conditions, the Dirichlet B.C. yields $-\frac{1}{4\pi^2} + c_2 = 0 \iff c_2 = \frac{1}{4\pi^2}$, and the Neumann B.C. thus yields $\frac{1}{2\pi}\sin(2\pi) + \frac{1}{2} + c_1 = 0 \iff c_1 = -\frac{1}{2}$, resulting in the analytical solution

$$u_a(x) = -\frac{1}{4\pi^2}\cos(2\pi x) + \frac{1}{6}x^3 - \frac{1}{2}x + \frac{1}{4\pi^2}$$

For the second boundary condition we get for c_2

$$u(0) = 1 \implies c_2 = 1 + \frac{1}{4\pi^2},$$

and hence for c_1:

$$c_1 + c_2 = 1 - \frac{1}{6} - \frac{1}{4\pi^2} \iff c_1 = -\frac{1}{6}$$

And so, by rearranging the terms, we get

$$u_b(x) = \frac{2\pi^2(x^3 - x + 6) - 3\cos(2\pi x) + 3}{12\pi^2}$$

1.1.1.2 Ill-posed BVP

For the last boundary condition, we may only determine c_1 analytically, so we have $u_x(0) = c_1 = 0 \iff u = \frac{-1}{4\pi^2}\cos(2\pi x) + \frac{1}{6}x^3 + c_2$. In addition, the finite difference scheme (as mentioned in 1.1.1.3) would become

$$\frac{1}{h^2}\begin{bmatrix} -h & h & 0 & \dots & 0 \\ 1 & -2 & 1 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 1 & -2 & 1 \\ 0 & \dots & 0 & h & -h \end{bmatrix}\begin{bmatrix} U_0 \\ U_1 \\ \vdots \\ \vdots \\ U_M \end{bmatrix} = \begin{bmatrix} \sigma_0 + \frac{h}{2}f_0 \\ f_1 \\ \vdots \\ f_{M-1} \\ \frac{h}{2}f_M - \sigma_1 \end{bmatrix},$$

which is singular, supporting the claim that the problem has infinitely many solutions.

A possible remedy for this issue is to impose a Dirichlet condition at both ends, e.g. $u(0) = 0$. Another possibility is to impose that $\sum_i f_i = 0$.

1.1.1.3 Discretization and difference schemes

With the Dirichlet-Neumann conditions (case a), we may approximate $u_x(1) = \sigma$ without the use of fictitious nodes with the discretization

$$\frac{-\frac{1}{2}u_{m-2} + 2u_{m-1} - \frac{3}{2}u_m}{h}$$

And so we will get the scheme

$$A_h U = F$$

$$\frac{1}{h^2}\begin{bmatrix} -2 & 1 & 0 & \dots & 0 \\ 1 & -2 & 1 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 1 & -2 & 1 \\ 0 & \dots & -\frac{h}{2} & 2h & -\frac{3h}{2} \end{bmatrix}\begin{bmatrix} U_0 \\ U_1 \\ \vdots \\ \vdots \\ U_M \end{bmatrix} = \begin{bmatrix} \sigma \\ f_1 \\ \vdots \\ f_{M-1} \\ f_M - \frac{\beta}{h^2} \end{bmatrix} \tag{4}$$

For the case with pure Dirichlet boundary conditions, we may use the discretization

$$\frac{1}{h^2}(u_{m-1} - 2u_m + u_{m+1}),$$

leading to the first-order scheme

$$\frac{1}{h^2}\begin{bmatrix} -2 & 1 & 0 & \dots & 0 \\ 1 & -2 & 1 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 1 & -2 & 1 \\ 0 & \dots & 0 & 1 & -2 \end{bmatrix}\begin{bmatrix} U_0 \\ U_1 \\ \vdots \\ \vdots \\ U_M \end{bmatrix} = \begin{bmatrix} f_0 - \frac{\alpha}{h^2} \\ f_1 \\ \vdots \\ f_{M-1} \\ f_M - \frac{\beta}{h^2} \end{bmatrix}, \tag{5}$$

where $\alpha = u(0), \beta = u(1)$.

The scheme in (4) is of order 2, which can be confirmed with the method of undetermined coefficients, in which, for $k = j + 1$,

$$\tau_n = \frac{h^k}{k!}u_n^{(k)}\sum_{l=p}^{q} a_l l^k + \mathcal{O}(h^k),$$

with $p = -2$, $q = 0$, $j = q - p = 2$, and $(a_{-2}, a_{-1}, a_0) = (-\frac{1}{2h}, \frac{2}{h}, -\frac{3}{2h})$, as in [5, p. 10]. We get

$$\tau_n = \frac{h^3}{3!}u_n^{(3)}\left(\frac{8}{2h} - \frac{2}{h}\right) = \frac{h^3(8-4)}{12h}u_n^{(3)} = \mathcal{O}(h^2)$$

Similarly, the schema in (5) is of order $\mathcal{O}(h)$.

1.1.2 Computational aspects

1.1.2.1 Sparse matrices

The main computational takeaway is the sparsity of the difference schemes. Since they are sparse, which we can utilize the SciPy Sparse module to drastically improve computation time. SciPy allows us to set up the difference schemes using e.g. NumPy and convert them to a Compressed Sparse Column (CSC) matrix to more efficiently find the LU-decomposition of the matrix A and the solution U thereof.

1.1.2.2 Continuous norms on numerical approximations

In order to analyze the convergence of the schemes, we implement discrete ℓ_2-norm and the continuous L_2-norm. The discrete ℓ_2-norm, defined by $\|V\|_2 = \sqrt{\frac{1}{N}\sum_{i=1}^{N} V_i^2}, V_i \in V$, is bundled with numpy through the function `numpy.linalg.norm`. A continuous L_2-norm, defined by $\|v(x)\|_2 = \sqrt{\int_\Omega v^2(x) d\Omega}$, is not as straight-forward since we initially only have approximations on specific points. The implementation is described in Algorithm 1.

Algorithm 1 Continuous L_2 norm for discrete points

1: **procedure** CONTINUOUS L_2 NORM
2: **input:** n-dimensional vector $\boldsymbol{V}$ and corresponding grid X.
3: $f(x) \leftarrow$ Linear spline interpolation of $\boldsymbol{V}$ onto X
4: $g(x) \leftarrow$ Anonymous function $f^2(x)$ that can be passed in an integrator
5:

$$res \leftarrow \sqrt{\int_{x_0}^{x_n} g(x)dx}, x \in X,$$

▷ Solve integral e.g. with Gaussian quadrature

6: **return** res

1.1.2.3 Adaptive Mesh Refinement

Another computational aspect is the implementation of the Adaptive Mesh Refinement (AMR) to the given pure Dirichlet problem. This is done by expressing the step length h itself as a grid (H), initially with non-varying value h_0 being the difference between two nodes on the uniform grid X. For each $x_i \in X$, we determine whether

$$\|U_{approx}(x_i) - u(x_i)\|_{L_2} > \alpha E, \tag{6}$$

where E is either

$$\max_i \|U_{approx}(x) - u(x)\|_{L_2} \text{ on } \Omega = [x_i, x_{i+1}] \text{ with } \alpha = 1, \text{ or} \tag{7}$$

$$\frac{\sum_{i=1}^{N} \|U_{approx}(x) - u(x)\|_{L_2}}{N} \text{ on } \Omega = [x_i, x_{i+1}] \text{ with } \alpha = 0.7. \tag{8}$$

Here, $U_{approx}(x_i)$ a cubic spline interpolation value at the grid point x_i, and $u(x_i)$ is the analytical (manufactured) solution at x_i.

For all x_i meeting the criterion, we refine the interval by adding the point $\frac{1}{2}(x_i + x_{i+1})$, e.g. splitting 0.4 on the grid $\{0.2, 0.4, 0.6, \dots\}$ would yield $\{0.2, 0.4, 0.5, 0.6, \dots\}$. This implementation is described in Algorithm 2. More technical specifications are available from the attached code.

1.1.2.4 Solving linear systems on non-uniform grids

In order to attain a solution to the system $AU = F$ on a non-uniform grid, a straight-forward solver is not sufficient. Instead, we use the following stencils, as described in [4]:

Algorithm 2 Adaptive Mesh Refinement for 1D Poisson equation

procedure DO_AMR
input: Grid X, refinement method, BVP-problem with analytical solution $u(x_i)$ and approximated solution U.
$U_{spline} \leftarrow$ cubic spline interpolation of U on X
for $x_i \in X$ **do**
Determine E based on (7) or (8)

if $\|U_{approx}(x_i) - u(x_i)\|_{L_2} > \alpha E$ **then**
Add point $\frac{1}{2}(x_i + x_{i+1})$ and store in X_{ref}.
$X \leftarrow X + X_{ref}$
Sort X
Update H s.t. $h_i = x_{i+1} - x_i \forall h_i \in H$
Set up difference schema and solve $AU = F$ with the stencil in section 1.1.2.4. **return** Approximated U from linear system

procedure REFINE UNTIL
Input: Grid X, refinement method, BVP-problem with analytical solution $u(x_i)$ and approximated solution U.
while Global error too large or Ndof < given limit **do**
$U \leftarrow$ DO_AMR
return Ndofs with corresponding relative error

Using $d_{i-2} = |x_{i-2} - x_i|$, $d_{i-1} = |x_{i-1} - x_i|$, $d_{i+1} = |x_i - x_{i+1}|$ and $\Delta x_i =$ for nodes x_i on the grid X, we use the discretization

$$(u_{xx})_i \approx \alpha u_{i-2} + \beta u_{i-1} - (\alpha + \beta + \gamma)u_i + \gamma u_{i+1}$$

The coefficients for order 2 are given by

$$\alpha = \frac{2(d_{i+1} - d_{i-1})}{d_{i-2}(d_{i-2} + d_{i+1})(d_{i-2} - d_{i-1})}$$
$$\beta = \frac{2(d_{i-2} - d_{i+1})}{d_{i-1}(d_{i-2} - d_{i-1})(d_{i-1} + d_{i+1})}$$
$$\gamma = \frac{2(d_{i-2} + d_{i-1})}{d_{i+1}(d_{i-1} + d_{i+1})(d_{i-2} + d_{i+1})}$$

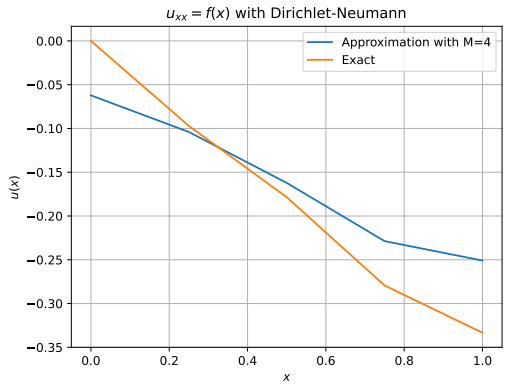
and for order 1, they are

$$\alpha = \frac{d_{i-1}d_{i+1}}{d_{i-2}(d_{i-2} + d_{i+1})(d_{i-2} - d_{i-1})}$$
$$\beta = \frac{-d_{i-2}d_{i+1}}{d_{i-1}(d_{i-2} - d_{i-1})(d_{i-1} + d_{i+1})}$$
$$\gamma = \frac{d_{i-2}d_{i-1}}{d_{i+1}(d_{i-1} + d_{i+1})(d_{i-2} - d_{i+1})}$$

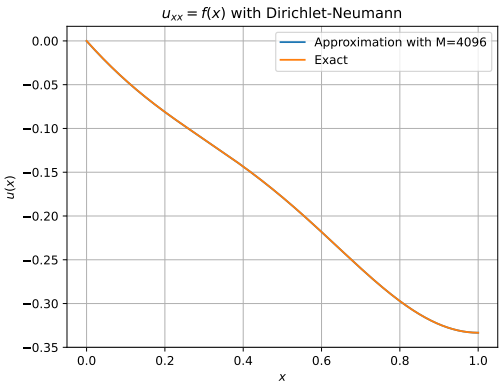
The coefficients are from [4, Table 1]. This is implemented by starting with a central difference and thereafter populating A in terms of the coefficients described above. We then insert the boundary conditions and solve for U.

1.1.3 Numerical examples

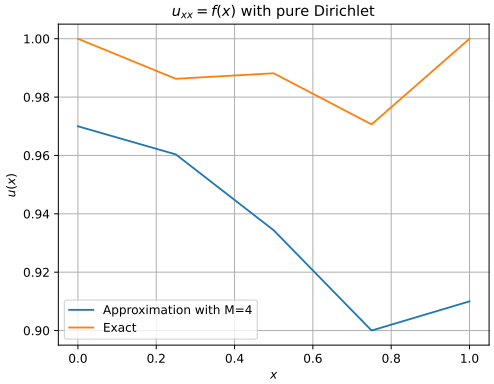
We start by solving the equation with $M = 2^2, 2^3, \ldots, 2^{12}$ and comparing the approximation between $M = 4$ and $M = 4096$ in Figure (1).



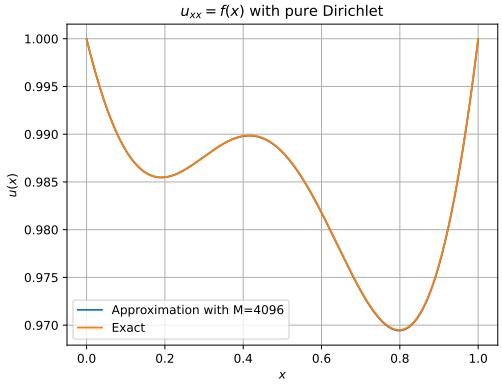
(a) $M = 4$ for Dirichlet-Neumann case



(b) $M = 4096$ for Dirichlet-Neumann case



(c) $M = 4$ for pure Dirichlet



(d) $M = 4096$ for pure Dirichlet

Figure 1: Analytical and approximated $u(x)$ from problem (1) with varying boundary conditions, $f(x)$ from (2).

We see that even for $M = 4$, the schema seem to catch some of the main contours of the plot, whereas for $M = 4096$, the analytical and approximated solution are indistinguishable.

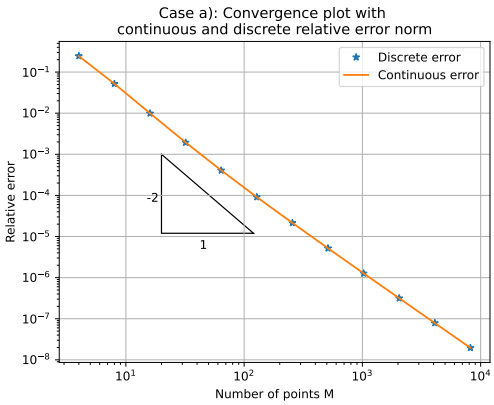
Furthermore, Figure 2 shows the relative (discrete and continuous) error for UMR (Uniform Mesh Refinement), for the Dirichlet-Neumann and pure Dirichlet case, respectively. These plots show a log-log ratio of the relative error, e_l^r and $e_{L_2}^r$ plotted against the number of internal nodes M. The discrete evaluations are at $M = \{2^k\}_{k=1}^{13}$. The relative errors are defined by

$$e_l^r := \frac{\|u - U\|_2}{\|u\|_2}$$
$$e_{L_2}^r := \frac{\|u(x) - U(x)\|_2}{\|u(x)\|_2},$$

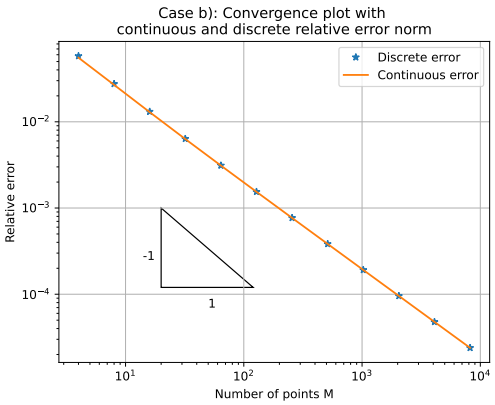
where $\|\cdot\|_{\ell_2}$ and $\|\cdot\|_{L_2}$ are defined in section 1.1.2.2. Implementation of the latter relative error is described in Algorithm 1.

As expected, the second-order scheme produces a log-log plot with a slope of -2, and the first-order scheme (for pure Dirichlet conditions) produces a log-log plot with slope of -1.

For the case of the manufactured solution from 3, the convergence plots are given in Figure 4, where we have chosen $\varepsilon = 1/1000$. We can see that for AMR of both orders 1 and 2, the relative error is of order $\mathcal{O}(h^2)$. However, since we are adaptively refining the grid we may experience convergence rates of orders higher than that of refining uniformly. As such, the expected convergence rates from [4, Table 1, order of accuracy] can be considered a lower bound performance-wise, and is thus not in conflict with the results.

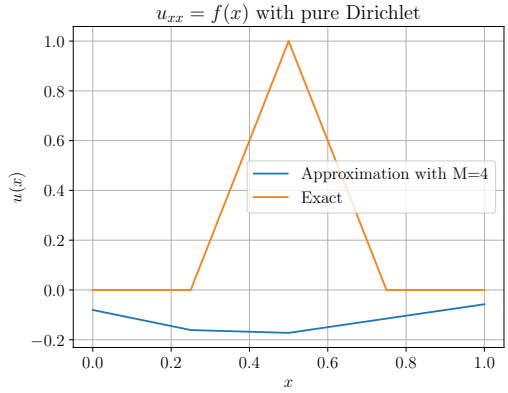


(a) Dirichlet-Neumann condition

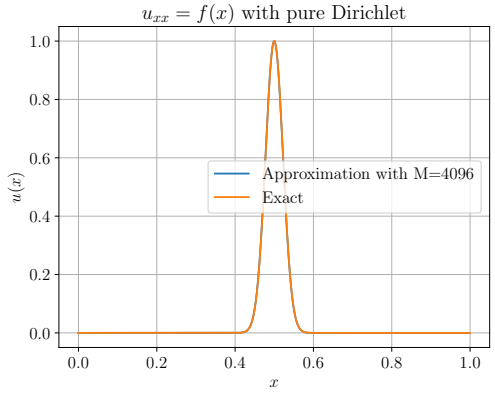


(b) Pure Dirichlet condition

Figure 2: Convergence plot for Poisson equation with varying boundary conditions.

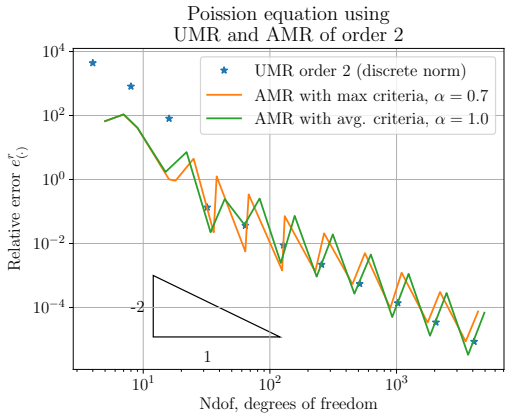


$N = 4$

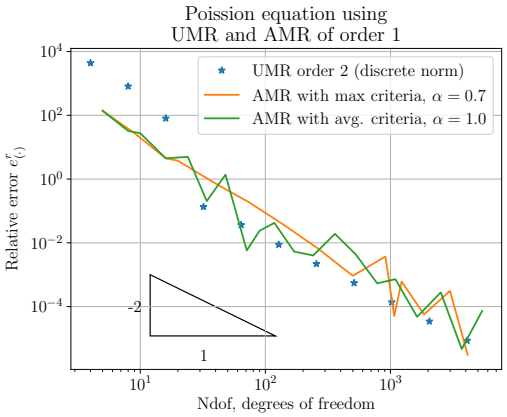


$N = 4096$

Figure 3: Plot of the approximation and analytical solution of the manufactured Poisson problem given in (3). Shown for both first and last step.



(a) Order 2.



(b) Order 1.

Figure 4: AMR on $u_{xx} = f(x)$ with manufactured solution (3).

1.2 Parabolic equations: Heat equation and inviscid Burgers' equation

1.2.1 Heat equation with Neumann BC

In this task we consider the heat equation with Neumann boundary conditions

$$\begin{cases} u_t = u_{xx}, \\ x \in [0,1],\ t > 0 \\ u_x(0,t) = u_x(1,t) = 0, \\ u(x,0) = 2\pi x - \sin 2\pi x \end{cases} \tag{9}$$

With Dirichlet boundary conditions, it is generally easy to obtain a method of order 2 in space for the heat equation. The central difference approximation of the derivative is of order 2, and so by using e.g. semidiscretization with central differences in space, as in [5] p. 36, all that is left is to choose an appropriate method for the resulting system of ODEs. However, the Von Neumann conditions introduce a new complication in that the values on the boundary aren't specified directly. Thus, a discretization of the spatial derivative on the boundary is needed in order to solve (9).

1.2.1.1 Semidiscretization

First we use semidiscretization to obtain the system of ODEs. We define a grid $x_m = mh$, where $h = 1/(M+1)$ and $m = 0, 1, \ldots, M+1$.

$$\begin{aligned} \partial_t u(x_m,t) &= \partial_x^2 u(x_m,t) \\ &= \frac{1}{h^2}\delta_x^2 u(x_m,t) + \varphi(x_m,t) \end{aligned} \tag{10}$$

where δ_x is the central difference operator in the x direction, and $\varphi(x_m,t) = -\frac{1}{12}h^2\partial_x^4 u(x_m,t) + \ldots$ is the truncation error. As in [5, p. 37], we define the approximations $v_m(t)$, $m = 0, \ldots, M+1$, of $u(x_m,t)$. From (10) we get

$$\dot{v}_m = \frac{1}{h^2}\delta_x^2 v_m, \quad 0 \le m \le M+1 \tag{11}$$

where the dot indicates a time derivative.

Equation (11) is problematic, as it has implicitly assumed grid lines in $x = -h$ and $x = 1+h$. However, using discretizations of the space derivative at the boundary, we can eliminate these grid lines.

We first construct a method using first order approximations. With backwards differences at x_0 and forward differences at x_{M+1}, we get the system

$$\begin{cases} \dot{v}_m = h^{-2}\delta_x^2 v_m, & 0 \le m \le M+1 \\ \dfrac{v_0 - v_{-1}}{h} = 0, \\ \dfrac{v_{M+2} - v_{M+1}}{h} = 0, \end{cases} \tag{12}$$

where $v_{-1}(t) = u(-h,t)$ and $v_{M+2}(t) = u(1+h,t)$. The bottom two equations both have an $\mathcal{O}(h)$ error, and they imply that $v_{-1} = v_0$ and $v_{M+2} = v_{m+1}$. Thus,

$$\begin{aligned} \dot{v}_0 &= h^{-2}(v_1 - 2v_0 + v_{-1}) \\ &= h^{-2}(v_1 - v_0) \end{aligned}$$

and

$$\begin{aligned} \dot{v}_{M+1} &= h^{-2}(v_{M+2} - 2v_{M+1} + v_M) \\ &= h^{-2}(-v_{M+1} + v_M). \end{aligned}$$

Substituting this into (12), we get

$$\begin{cases} \dot{v}_m = h^{-2}\delta_x^2 v_m, \quad 1 \le m \le M \\ \dot{v}_0 = h^{-2}(v_1 - v_0), \\ \dot{v}_{M+1} = h^{-2}(-v_{M+1} + v_M). \end{cases} \tag{13}$$

Defining $\vec{v} = [v_0, \ldots, v_{M+1}]^T$, we can write (13) in matrix format,

$$\dot{\vec{v}} = \frac{1}{h^2}\begin{bmatrix} -1 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -1 \end{bmatrix}\vec{v},$$

or

$$\dot{\vec{v}} = \frac{1}{h^2}A\vec{v}. \tag{14}$$

This will be the basis for our first method, and it can be solved with standard ODE procedures. The discretization error is $\mathcal{O}(h)$ because of the first order approximations at the boundaries.

Secondly, we do the same thing, except for using second order approximations on the boundary. With central differences, the bottom two equations in (12) are replaced by

$$\frac{v_1 - v_{-1}}{2h} = 0,$$
$$\frac{v_{M+2} - v_M}{2h} = 0.$$

Thus,

$$\dot{v}_0 = 2h^{-2}(v_1 - v_0),$$
$$\dot{v}_{M+1} = 2h^{-2}(-v_{M+1} + v_M),$$

and our method becomes

$$\dot{\vec{v}} = \frac{1}{h^2}\begin{bmatrix} -2 & 2 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 2 & -2 \end{bmatrix}\vec{v},$$

or

$$\dot{\vec{v}} = \frac{1}{h^2}B\vec{v}. \tag{15}$$

This will be the basis for our second method. Now the error in both the boundary approximations and spatial discretization is $\mathcal{O}(h^2)$, so the total discretization error is $\mathcal{O}(h^2)$.

Solving these systems of ODE's with the trapezoidal method gives the Crank Nicholson method. Assume $\vec{y}_m(t)$ is a solution to the equation

$$\dot{\vec{y}}_m = \vec{F}(t, \vec{y}_m).$$

Then the trapezoidal rule yields

$$\vec{y}_m^{n+1} := \vec{y}_m(t_{n+1}) = \vec{y}_m^n + \frac{k}{2}\Big(\vec{F}(t_n, \vec{y}_m^n) + \vec{F}(t_{n+1}, \vec{y}_m^{n+1})\Big) + \psi_m^n, \tag{16}$$

with $\psi_m^n = -\frac{1}{12}k^3\partial_t^3 y_m(t_n) + \ldots$. By plugging (14) and (15) into (16) and rearranging, our two methods for solving problem (9) becomes

$$\Big(I - \frac{r}{2}A\Big)\vec{v}^{n+1} = \Big(I + \frac{r}{2}A\Big)\vec{v}^n, \tag{17}$$
$$\Big(I - \frac{r}{2}B\Big)\vec{v}^{n+1} = \Big(I + \frac{r}{2}B\Big)\vec{v}^n \tag{18}$$

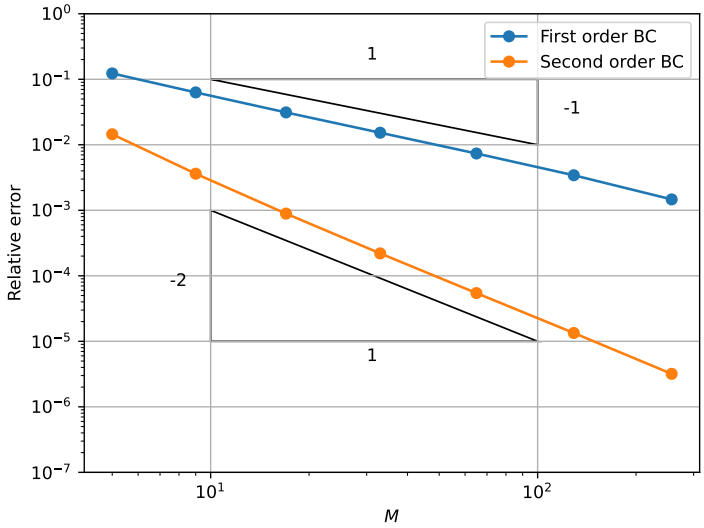


Figure 5: Convergence of Crank Nicholson with first and second order boundary conditions applied to problem (9) at $t = 0.1$ with $k = 10^{-4}$.

Here, (17) uses a first order approximation for the boundary conditions, while (18) uses a second order approximation.

From (10) we know that $\partial_t u(x_m, t) = 1/h^2\, \delta_x^2 u(x_m, t) + \varphi_m(t)$. Substituting this into (16) gives

$$\begin{aligned} u_m^{n+1} &= u_m^n + \frac{k}{2}\left(\frac{1}{h^2}\delta_x^2 u_m^n + \varphi_m^n + \frac{1}{h^2}\delta_x^2 u_m^{n+1} + \varphi_m^{n+1}\right) + \psi_m^n \\ &= u_m^n + \frac{k}{2}\left(\frac{1}{h^2}\delta_x^2 u_m^n + \frac{1}{h^2}\delta_x^2 u_m^{n+1}\right) + k\tau_m^n, \end{aligned}$$

with $k\tau_m^n = \frac{k}{2}(\varphi_m^n + \varphi_m^{n+1}) + \psi_m^n$. Using this to solve (14) gives $\tau_m^n = \mathcal{O}(h + k^2)$, while method (15) yields $\tau_m^n = \mathcal{O}(h^2 + k^2)$.

Let $M = 1/h - 1$ be the number of points in the interior of our spatial discretization. Using a sufficiently small value for k, we thus expect (14) and (15) to be $\mathcal{O}(M^{-1})$ and $\mathcal{O}(M^{-2})$ respectively. The convergence plot confirming this is shown in figure 5, where the error for each M have been computed by comparing the solution to a reference solution with $M^* = 1024$.

Manufactured solution and mesh refinement

Here we consider the heat equation on the rectangle $\Omega : \; x \in [0, 1]$, $t \in [0, T]$. However, we will now use the manufactured solution

$$u(x, t) = e^{-4\pi^2 t} \sin\left(2\pi x - \frac{\pi}{2}\right). \tag{19}$$

This is plotted in Equation (19) solves the initial/boundary value problem

$$\begin{cases} u_t = u_{xx} \text{ in } \Omega \\ \partial_x u(0, t) = \partial_x u(1, t) = 0 \\ u(x, 0) = \sin(2\pi x - \pi/2) \end{cases} \tag{20}$$

We will solve this problem using the backward Euler and trapezoidal methods for ODEs on system (15), corresponding to the backward Euler (BE) and Crank Nicholson (CN) methods for PDEs respectively. First, convergence will be tested by using uniform mesh refinement (UMR) in the x

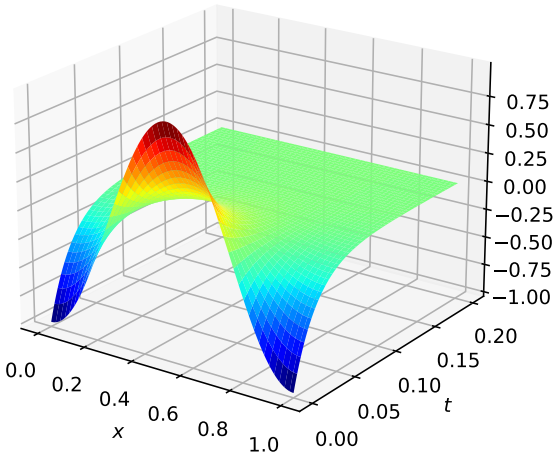


Figure 6: Plot of $u(x, y)$ as defined in (19). Here $T = 0.2$.

and t directions. Then we will investigate convergence under mesh refinement when $k = ch$ and $k = rh^2$, where c and r are positive constants.

In order to measure the error, it is possible to use either a discrete or continuous norm. For the discrete norm, we can use a relative measure with the traditional ℓ_2 norm,

$$e_\ell^r = \frac{\sqrt{\sum_{i=0}^{M+1} \frac{1}{M+2}(u(x_i, T) - V_i)^2}}{\sqrt{\sum_{i=0}^{M+1} \frac{1}{M+2} u(x_i, T)^2}}, \tag{21}$$

where $u(x_i, T)$ is the exact solution in $(x_i, T) \in \Omega$ and V_i is the i^{th} component of the approximated solution $\vec{V}$ at time T. For the continuous norm, we can use a relative version of the traditional L_2 norm. In order to use this, however, we must first interpolate a cubic spline to the nodes in $\vec{V}$, getting a continuous function $v_c(x)$. The error function becomes

$$e_L^r = \frac{\sqrt{\int_X (u(x, T) - v_c(x))^2 \mathrm{d}x}}{\sqrt{\int_X u(x_i, T)^2 \mathrm{d}x}}, \tag{22}$$

where $X = [0, 1]$. All methods will be measured against the manufactured solution (19) at time $T = 0.2$.

In the last task we derived that the CN method has discretization error $\mathcal{O}(h^2 + k^2)$. With the same argument it is possible to show that the BE method has discretization error $\mathcal{O}(h^2 + k)$. When k is a small constant, we thus expect both CN and BE to be of order 2 in space. However, the noise introduced by k should become significant faster in BE than in CN as h decreases. Fixing $k = T \cdot 10^{-4}$, we reduce noise enough to see the spatial convergence of the methods. The error was computed using both types of error function at $h = 2^s$, $s = 2, \ldots, 12$. This is plotted in figure 7. As was predicted, both methods is of order 2 in space, even though BE gets saturated faster.

Similarly, when h is a small constant, we expect CN to be of order 2 in time, while BE is of order 1. The error introduced by h is small compared to that introduced by k previously, since both CN and BE are of order 2 in space. However, CN will still start to saturate at a relative error of roughly 10^{-5}. We therefore fix $h = 10^{-4}$. The error was computed using both types of error

function at $k = 2^s$, $s = 4, \ldots, 13$. This is plotted in figure 8. The methods follow the predicted orders.

Convergence was also explored using refinement of a grid with $k = ch$, where c is a positive constant. Substituting $k = ch$ into the order expressions of the methods, we find that CN is $\mathcal{O}(h^2)$ and BE is $\mathcal{O}(h)$. However, since both k and h are refined simultaneously, it makes sense to introduce the number of degrees of freedom, Ndof $= MN$, as a parameter. Here M, N are the number of node points in the x and t directions respectively. Using this, we find that the orders of the methods are $\mathcal{O}(\text{Ndof}^{-1})$ and $\mathcal{O}(\text{Ndof}^{-0.5})$ for CN and BE respectively. Since both methods have a higher order of convergence in space than in time, it makes sense to let c be small. Figure 9 shows the convergence of the two methods with $c = 1/100$, using Ndof $= 100 \cdot 2^{2s-1}$, $s = 5, \ldots, 11$. The methods follow the expected rates.

Lastly, convergence was explored while requiring $k = rh^2$ on the grid, where r is some positive constant. By [5] p. 56, both methods are stable for all $r \geq 0$, we may set $r = 0.5$. Using the grid requirement together with the Ndof parameter, it is easy to show that both CN and BE becomes $\mathcal{O}(\text{Ndof}^{\frac{2}{3}})$. CN should, however, maintain a lower error than BE. All this is confirmed in figure 10, where the relative error is plotted for Ndof $= 2^{3s-1}$, $s = 4, \ldots, 10$.

1.2.2 Inviscid Burgers equation

We now consider the inviscid Burgers equation on $x \in [0, 1]$ with $t > 0$,

$$\begin{cases} u_t = -uu_x \\ u(0,t) = u(1,t) = 0 \\ u(x,0) = \exp\left(-400(x - 1/2)^2\right) \end{cases} \tag{23}$$

We consider the grid $x_m = mh$, $m = 0, \ldots, M+1$ with $h = \frac{1}{M+1}$, and introduce $v_m(t) \approx u(x_m, t)$. Using central differences in space, we get the following semidiscretized equation.

$$\begin{cases} v_0(t) = v_{M+1}(t) = 0 \\ \dot{v}_m = -\frac{1}{2h} v_m (v_{m+1} - v_{m-1}), \quad m = 1, \ldots, M \end{cases} \tag{24}$$

This system can be solved with the function `solve_ivp` from the `scipy` library in Python. The solution for three different time points is shown in figure 11. Figure 11c shows that the numerical solution breaks at time $t^* \approx 0.06$.

1.3 Elliptic equations: 2D Laplace equation

In this part we will consider the 2D Laplace equation on the unit square with given boundary conditions. First we solve the problem analytically using separation of variables, then we implement the 5-point formula and give convergence plots that show the order of convergence.

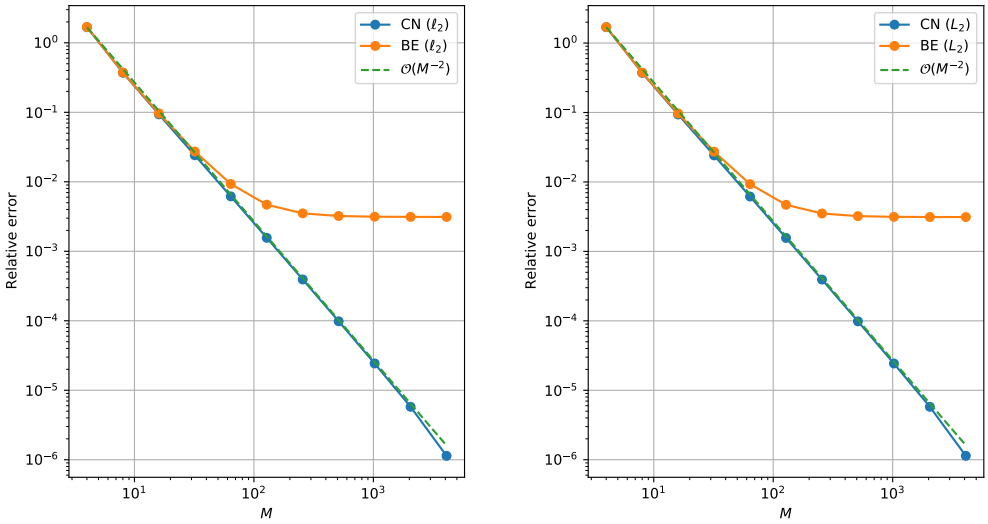
1.3.1 Analytical solution to 2D Laplace equation on unit square

We consider the 2D Laplace equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0, \tag{25}$$

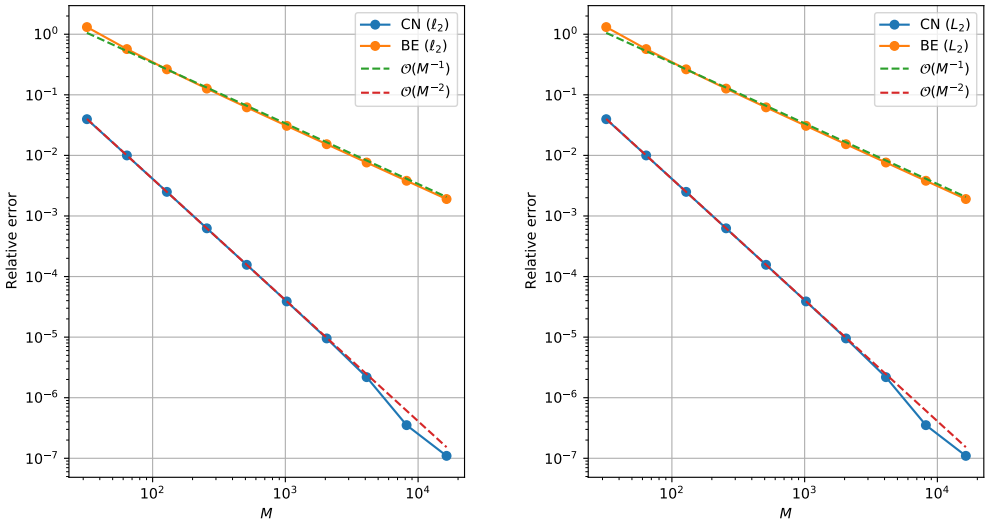
on the unit square $(x, y) \in [0, 1]^2$, where the boundary is given as

$$u(x, y) = g(x, y),$$



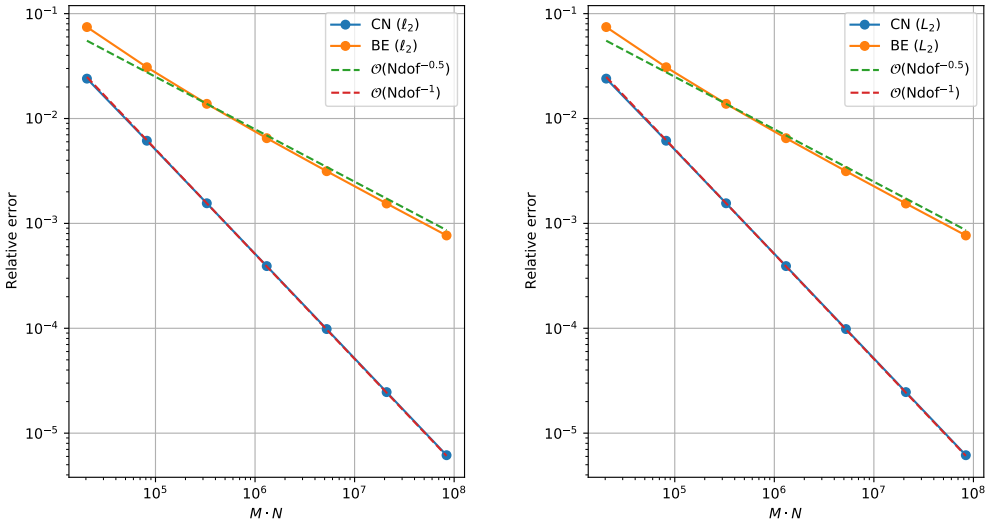
(a) Discrete norm (b) Continuous norm

Figure 7: Convergence of the solutions of (20) with discrete (ℓ_2) and continuous (L_2) norms while refining h with $k = 10^{-4}$ fixed, $T = 0.2$.



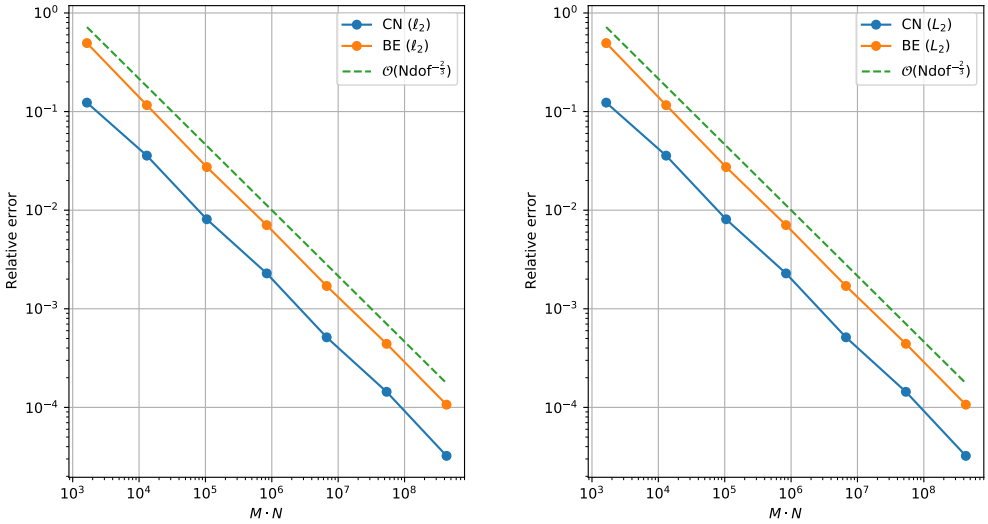
(a) Discrete norm (b) Continuous norm

Figure 8: Convergence of the solutions of (20) with discrete (ℓ_2) and continuous (L_2) norms while refining k with $h = 10^{-4}$ fixed, $T = 0.2$.



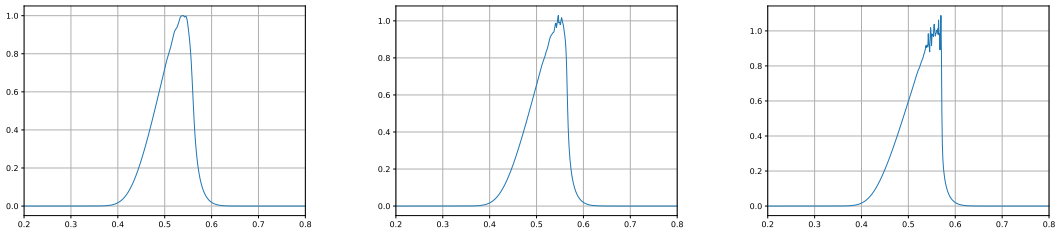
(a) Discrete norm (b) Continuous norm

Figure 9: Convergence of the solutions of (20) with discrete (ℓ_2) and continuous (L_2) norms while refining h with $k = \frac{h}{100}$, $T = 0.2$.



(a) Discrete norm (b) Continuous norm

Figure 10: Convergence of the solutions of (20) with discrete (ℓ_2) and continuous (L_2) norms while refining h with $k = 0.5h^2$, $T = 0.2$.



(a) $t = 0.04$ (b) $t = 0.05$ (c) $t = 0.06$

Figure 11: Solution of problem (23) at different time points.

and has the following boundary conditions:

$$g(0, y) = 0, \qquad 0 \leq y \leq 1, \tag{26}$$
$$g(x, 0) = 0, \qquad 0 \leq x \leq 1, \tag{27}$$
$$g(1, y) = 0, \qquad 0 \leq y \leq 1, \tag{28}$$
$$g(x, 1) = \sin(2\pi x), \qquad 0 \leq x \leq 1. \tag{29}$$

First we use separation of variables, which is possible if we assume the solution can be written as a product of functions that only depend on one variable, i.e. we assume $u(x, y) = X(x)Y(y) = XY$.

Thus, the Laplace equation (25) can be rewritten as

$$\begin{aligned} X''Y + XY'' &= 0 \\ \implies \frac{X''}{X} &= -\frac{Y''}{Y} = -k \\ \implies \frac{X''}{X} &= -k, \quad \frac{Y''}{Y} = k \end{aligned}$$

Now that we have two systems of ordinary differential equations (ODE), we solve:

$$X'' + kX = 0, \quad X(0) = 0, X(1) = 0 \tag{30}$$
$$Y'' - kY = 0, \quad Y(0) = 0 \tag{31}$$

Boundary conditions (26) and (28) gives $X(0) = 0,\ X(1) = 0$, and we have $k_n = (n\pi)^2$ and the corresponding nonzero solutions

$$X(x) = X_n(x) = \sin(n\pi x),\ n = 1, 2, 3, ... \tag{32}$$

We can then rewrite (31) as

$$Y'' - (n\pi)^2 Y = 0,\ Y(0) = 0,$$

yielding the general solution

$$Y(y) = C_1 \cosh(n\pi y) + C_2 \sinh(n\pi y).$$

From (27) we know that the general solution has to satisfy $Y(0) = 0$, therefore $C_1 = 0$, and so we have

$$Y(y) = C_2 \sinh(n\pi y). \tag{33}$$

We can now make a multiply the two solutions (32) and (33) for the two ODEs and we have

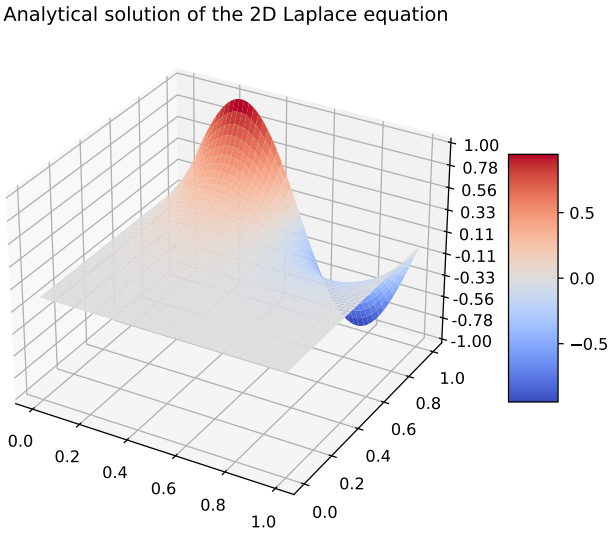


Figure 12: Surface plot of the analytical solution for the Laplace equation on $[0,1]^2$ with given boundary conditions (25).

$$u_n(x,y) = B_n \sinh(n\pi y)\sin(n\pi y),\ n = 1,2,3,...$$

We then sum over all these solutions and we have

$$u(x,y) = \sum_{n=1}^{\infty} B_n \sinh(n\pi y)\sin(n\pi x). \tag{34}$$

Now we apply the boundary condition (29), we need to express (34) as

$$\sin(2\pi x) = \sum_{n=1}^{\infty} B_n \sinh(n\pi)\sin(n\pi x),$$

which is possible if $B_2 = \frac{1}{\sinh(2\pi)}$ and $B_n = 0$ for all $n \neq 2$.

This results in the analytical solution

$$u(x,y) = \sum_{n=1}^{\infty} B_n \sinh(n\pi y)\sin(n\pi x) = \frac{1}{\sinh(2\pi)}\sinh(2\pi y)\sin(2\pi x) \tag{35}$$

Figure 12 contains a surface plot of the analytical solution (35).

1.3.2 5-point formula

To numerically solve (25), we can use the five point formula. We discretize our domain, i.e. $[0,1]^2$ with square sizes $h \times k$. h is then the step-size in x-direction and k is the step-size in the y-direction. The 5-point stencil of a point (x,y) in the grid is

$$\{(x-h,y),(x,y),(x+h,y),(x,y+k),(x,y-k)\}.$$

This is used to approximate the second order derivatives. The Taylor-expansions of u_{xx} and u_{yy} are

$$\begin{aligned} u_{xx}(x,y) &= \frac{u(x-h,y)+u(x+h,y)-2u(x,y)}{h^2} - 2\frac{u^{(4)}(x,y)}{4!}h^2 + \dots, \\ u_{yy}(x,y) &= \frac{u(x,y-k)+u(x,y+k)-2u(x,y)}{k^2} - 2\frac{u^{(4)}(x,y)}{4!}k^2 + \dots. \end{aligned} \tag{36}$$

We see that the points that are evaluated by u in (36) corresponds to the points of the 5-point stencil if we approximate the functions by only considering the first terms. Therefore we have a local truncation error of $\mathcal{O}(h^2+k^2)$.

1.3.3 Convergence plots using the 5-point formula

Figure 13 shows the discretization error, both in x-direction, y-direction, and the total error, i.e. discretization in both directions simultaneously.

The discretization error is defined, asymptotically, as

$$\frac{1}{\sqrt{kh}}\|u - U_{approx}\|_{\ell_2}$$

Since we from (36) should expect the scheme to converge at order $\mathcal{O}(h^2)$ and $\mathcal{O}(k^2)$, we should, based on the scaling factor $\frac{1}{\sqrt{kh}}$, expect $\mathcal{O}(h)$ in both directions. Although there is some deviance at the end in the decomposed errors, the rate is indeed $\mathcal{O}(Ndof^{-1})$ for all plots.

1.4 Hyperbolic equations: Linearized Korteweg-deVries

We are to analyze the linearized Korteweg-deVries (KdV) equation for $x \in [-1,1]$, $t > 0$,

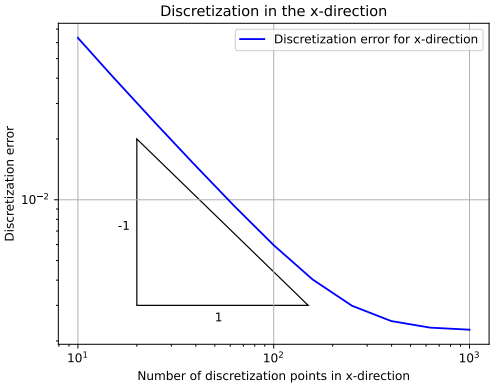
$$\begin{aligned} & u_t + \left(1+\pi^2\right) u_x + u_{xxx} = 0, \quad \text{with } u(x,0) = \sin(\pi x) \\ \implies\ & u_t = -\left(1+\pi^2\right) u_x - u_{xxx} \\ \text{where}\quad & u(x+2,t) = u(x,t). \end{aligned}$$

1.4.1 Discretization and stability

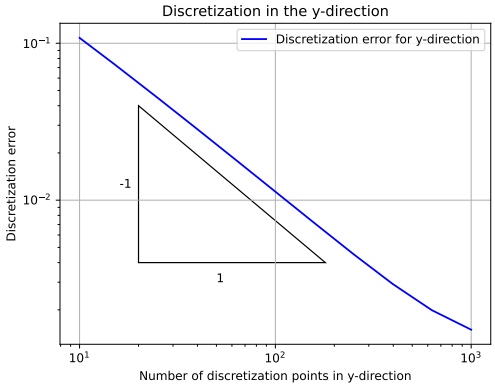
We fist discretize the equation in the space-direction using the central differences $u_x|_{x=x_m} = \partial_x u_m = (u_{m+1} - u_{m-1})/2h$ and $u_{xxx}|_{x=x_m} = \partial_{xxx} u_m = (u_{m+3} - 3u_{m+1} + 3u_{m-1} - u_{m-3})/8h^3$.

$$\begin{aligned} & u_t + \left(1+\pi^2\right) u_x + u_{xxx} = 0 \\ \text{which gives } & u_t + \left(1+\pi^2\right)\frac{u_{m+1}-u_{m-1}}{2h} + \frac{u_{m+3}-3u_{m+1}+3u_{m-1}-u_{m-3}}{8h^3} + O(h^2) = 0 \\ \implies\ & u_t = -\left(1+\pi^2\right)\frac{u_{m+1}-u_{m-1}}{2h} - \frac{u_{m+3}-3u_{m+1}+3u_{m-1}-u_{m-3}}{8h^3} + O(h^2). \end{aligned}$$

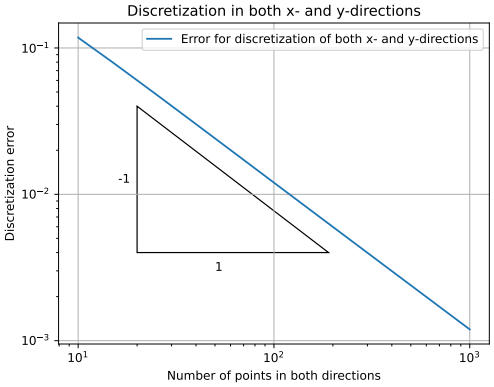
We note that for our spatial discretization $\mathbf{U}^n = [U_0^n, U_1^n, ..., U_{M-1}^n]$ with M points, we have a periodic boundary condition yielding $u_m^n = u_{m+M}^n$ which we can use in our schemes.



(a) x-direction



(b) y-direction



(c) Both directions

Figure 13: Convergence plots for the implementation of the 5-point formula numerically solving the 2D Laplace equation (25), with the given boundary conditions. The convergence plots show the discretization error in x-direction (a), y-direction (b) and for discretization of both x and y simultaneously (c). We see that the order of convergence is 1 as we would expect from the local truncation error.

1.4.1.1 Forwards Euler

The forwards Euler discretization for this problem is

$$\begin{aligned}
u_m^{n+1} =& u(x_m, t_n + k) \\
=& u_m^n + k\partial_t u_m^n + O(k^2) \\
=& u_m^n + k\left[-\left(1+\pi^2\right)u_x^n - u_{xxx}^n\right] + O(k^2) \\
=& u_m^n + k\left[-\left(1+\pi^2\right)\frac{u_{m+1}^n - u_{m-1}^n}{2h} - \frac{u_{m+3}^n - 3u_{m+1}^n + 3u_{m-1}^n - u_{m-3}^n}{8h^3} + O(h^2)\right] + O(k^2) \\
=& u_m^n + \frac{k}{8h^3}\left[-u_{m+3}^n + (3 - 4h^2(1+\pi^2))(u_{m+1}^n - u_{m-1}^n) + u_{m-3}^n\right] + O(kh^2 + k^2)
\end{aligned}$$

which after discarding the terms of order k^2 and kh^2 gives our approximation scheme

$$\implies U_m^{n+1} = U_m^n + \frac{k}{8h^3}\left[-U_{m+3}^n + (3 - 4h^2(1+\pi^2))(U_{m+1}^n - U_{m-1}^n) + U_{m-3}^n\right].$$

We evaluate the von Neumann stability of this scheme by inserting $U_m^n = \xi^n e^{i\beta x_m}$, letting $r = k/h^3$ and noting that $x_{m+1} = x_m + h$, which gives

$$\begin{aligned}
U_m^{n+1} &= U_m^n + \frac{r}{8}\left[-U_{m+3}^n + (3 - 4h^2(1+\pi^2))(U_{m+1}^n - U_{m-1}^n) + U_{m-3}^n\right] \\
\implies \xi^{n+1}e^{i\beta x_m} &= \xi^n e^{i\beta x_m} + \frac{r}{8}[-\xi^n e^{i\beta x_{m+3}} + (3 - 4h^2(1+\pi^2))(\xi^n e^{i\beta x_{m+1}} - \xi^n e^{i\beta x_{m-1}}) \\
&\qquad + \xi^n e^{i\beta x_{m-3}}] \\
\implies \xi &= 1 + \frac{r}{8}\left[-e^{i\beta 3h} + (3 - 4h^2(1+\pi^2))(e^{i\beta h} - e^{-i\beta h}) + e^{-i\beta 3h}\right] \\
&= 1 + \frac{r}{8}\cdot 2i\left[-\sin(\beta 3h) + (3 - 4h^2(1+\pi^2))\sin(\beta h)\right] \\
\implies |\xi|^2 &= 1^2 + \left(\frac{r}{4}\left[-\sin(\beta 3h) + (3 - 4h^2(1+\pi^2))\sin(\beta h)\right]\right)^2 \\
\implies \max_{\beta\in\mathbb{R}}|\xi|^2 &= 1 + \left(\frac{r}{4}\left[-(-1) + (3 - 4h^2(1+\pi^2))\right]\right)^2 \quad \left(\text{when } \beta = \frac{\pi}{2h}\right) \\
&= 1 + \left(\frac{r}{4}\left[4 - 4h^2(1+\pi^2)\right]\right)^2 \\
&= 1 + r^2\left[1 - h^2(1+\pi^2)\right]^2.
\end{aligned}$$

Now, assuming a least certain fineness in the spatial direction; $M \geq 10$, we get $h \leq \frac{2}{10} = \frac{1}{5} \implies h^2(1+\pi^2) \leq \frac{11}{25} < \frac{1}{2} \implies [1 - h^2(1+\pi^2)] > \frac{1}{2}$. Using this we have

$$\max_{\beta\in\mathbb{R}}|\xi|^2 = 1 + r^2\left[1 - h^2(1+\pi^2)\right]^2 \geq 1 + \frac{r^2}{4} > 1$$

but also

$$\begin{aligned}
&\max_{\beta\in\mathbb{R}}|\xi|^2 = 1 + r^2\left[1 - h^2(1+\pi^2)\right]^2 \leq 1 + r^2\cdot 1 \\
&\implies |\xi| \leq \sqrt{1+r^2} \leq 1 + \frac{r^2}{2} = 1 + k\frac{k}{2h^6} = 1 + k\mu, \quad \mu = \frac{k}{2h^6}.
\end{aligned}$$

This shows that, given that we keep $\mu \propto \frac{k}{h^6}$ constant, the forward euler method is von Neumann stable. See for example Figures 14 and 15 for numerical demonstrations of this stability dependent on μ.

1.4.1.2 Crank-Nicolson:

The Crank-Nicolson discretization for this problem is

$$\begin{aligned}
u_m^{n+1} =& u(x_m, t_n + k/2) \\
&= u_m^n + k\partial_t u_m^{n+1/2} + O(k^3) \\
&= u_m^n + \frac{k}{2}\left[-\left(1+\pi^2\right) u_x^n - u_{xxx}^n\right] + \frac{k}{2}\left[-\left(1+\pi^2\right) u_x^{n+1} - u_{xxx}^{n+1}\right] + O(k^3) \\
\implies u_m^{n+1} - \frac{k}{2}&\left[-\left(1+\pi^2\right)\frac{u_{m+1}^{n+1} - u_{m-1}^{n+1}}{2h} - \frac{u_{m+3}^{n+1} - 3u_{m+1}^{n+1} + 3u_{m-1}^{n+1} - u_{m-3}^{n+1}}{8h^3} + O(h^2)\right] \\
&= u_m^n + \frac{k}{2}\left[-\left(1+\pi^2\right)\frac{u_{m+1}^{n} - u_{m-1}^{n}}{2h} - \frac{u_{m+3}^{n} - 3u_{m+1}^{n} + 3u_{m-1}^{n} - u_{m-3}^{n}}{8h^3} + O(h^2)\right] \\
&\qquad\qquad + O(k^3) \\
\implies u_m^{n+1} + \frac{k}{16h^3}&\left[u_{m+3}^{n+1} + (3 - 4h^2(1+\pi^2))(-u_{m+1}^{n+1} + u_{m-1}^{n+1}) - u_{m-3}^{n+1}\right] \\
&= u_m^n + \frac{k}{16h^3}\left[-u_{m+3}^{n} + (3 - 4h^2(1+\pi^2))(u_{m+1}^{n} - u_{m-1}^{n}) + u_{m-3}^{n}\right] + O(kh^2 + k^3)
\end{aligned}$$

which again after discarding terms of order k^3 and kh^2 gives our approximation scheme

$$\begin{aligned}
\implies U_m^{n+1} + \frac{k}{16h^3}&\left[U_{m+3}^{n+1} + (3 - 4h^2(1+\pi^2))(-U_{m+1}^{n+1} + U_{m-1}^{n+1}) - U_{m-3}^{n+1}\right] \\
&= U_m^n + \frac{k}{16h^3}\left[-U_{m+3}^{n} + (3 - 4h^2(1+\pi^2))(U_{m+1}^{n} - U_{m-1}^{n}) + U_{m-3}^{n}\right].
\end{aligned}$$

Evaluating the von Neumann stability as described for the Euler method, this time immediately dividing by $\xi^n e^{i\beta x_m}$ after inserting $U_m^n = \xi^n e^{i\beta x_m}$, we get

$$\begin{aligned}
U_m^{n+1} + \frac{k}{16h^3}&\left[U_{m+3}^{n+1} + (3 - 4h^2(1+\pi^2))(-U_{m+1}^{n+1} + U_{m-1}^{n+1}) - U_{m-3}^{n+1}\right] \\
&= U_m^n + \frac{k}{16h^3}\left[-U_{m+3}^{n} + (3 - 4h^2(1+\pi^2))(U_{m+1}^{n} - U_{m-1}^{n}) + U_{m-3}^{n}\right] \\
\implies \xi + \frac{r}{16}\xi&\left[e^{i\beta 3h} + (3 - 4h^2(1+\pi^2))(-e^{i\beta h} + e^{-i\beta h}) - e^{-i\beta 3h}\right] \\
&= 1 + \frac{r}{16}\left[-e^{i\beta 3h} + (3 - 4h^2(1+\pi^2))(e^{i\beta h} - e^{-i\beta h}) + e^{-i\beta 3h}\right].
\end{aligned}$$

Defining $f(\beta, h)$ by

$$\begin{aligned}
2i \cdot f(\beta, h) =& \left[-e^{i\beta 3h} + (3 - 4h^2(1+\pi^2))(e^{i\beta h} - e^{-i\beta h}) + e^{-i\beta 3h}\right] \\
&= 2i\left[-\sin(\beta 3h) + (3 - 4h^2(1+\pi^2))\sin(\beta h)\right].
\end{aligned}$$

we get $f(\beta, h) \in \mathbb{R}$ and

$$
\begin{aligned}
\xi + \frac{r}{16}\xi(-2i)f(\beta,h) &= 1 + \frac{r}{16}2if(\beta,h) \\
\implies \xi &= \frac{1 + ir/8 \cdot f(\beta,h)}{1 - ir/8 \cdot f(\beta,h)} \\
\implies |\xi| &= \left|\frac{1 + ir/8 \cdot f(\beta,h)}{1 - ir/8 \cdot f(\beta,h)}\right| = \frac{|1 + ir/8 \cdot f(\beta,h)|}{|1 - ir/8 \cdot f(\beta,h)|} = \frac{\sqrt{1 + (r/8 \cdot f(\beta,h))^2}}{\sqrt{1 + (r/8 \cdot f(\beta,h))^2}} = 1.
\end{aligned}
$$

So, since $|\xi| \leq 1$, the method is stable.

1.4.2 Implementation and convergence plots

While the implementation is done in the proj4.py-file, the implementation uses a symmetry in our system for efficient solving. The basis of this method will be outlined here.

1.4.2.1 Circulant matrices Observing our schemes we find that since we have no boundary conditions other than periodicity ($U^n_{m+M} = U^n_m$), we will find that we can express our schemes as $\mathbf{U}^{n+1} = C\mathbf{U}^n$ where $\mathbf{U}^n$ is our vector of function values in state n and C is a $m \times m$ circulant matrix

$$
C = \begin{bmatrix}
c_0 & c_{m-1} & \dots & c_2 & c_1 \\
c_1 & c_0 & c_{m-1} & & c_2 \\
\vdots & c_1 & c_0 & \ddots & \vdots \\
 & & \ddots & \ddots & c_{m-1} \\
c_{m-2} & & & & \\
c_{m-1} & c_{m-2} & \dots & c_1 & c_0
\end{bmatrix}. \tag{37}
$$

where we define $\mathbf{c} = [c_0, c_1, \dots, c_{m-1}]$.

What is worth noting about circulant matrices for our implementation, is that they are diagonalized by equation (3) in [2, p. 2], the Fourier matrix F_m, and has eigenvalues $\lambda_1, \dots, \lambda_m$ calculated by a Fourier transform $F_m\mathbf{c}$:

$$
C = \frac{1}{m}F_m^* D_c F_m \quad \text{where } D_c = \operatorname{diag}(F_m\mathbf{c}) = \operatorname{diag}(\lambda_1, \lambda_2, \dots, \lambda_m). \tag{38}
$$

This also implies that the eigenvalues of the matrix are found by a discrete Fourier transform, in particular $F_m\mathbf{c} = DFT(\mathbf{c})$.

Furthermore, utilizing $F_m^* F_m = m \cdot I$, the following relations thus hold:

$$
\begin{aligned}
C^n &= \left(\frac{1}{m}F_m^* D_c F_m\right)^n = \frac{1}{m}F_m^* D_c^n F_m. \\
C \text{ circulant and invertible} &\implies C^{-1} = \frac{1}{m}F_m^* \operatorname{diag}(\lambda_1^{-1}, \dots, \lambda_m^{-1})F_m = \frac{1}{m}F_m^* D_c^{-1} F_m \text{ circulant.} \\
A, B \text{ circulant} &\implies AB = \frac{1}{m}F_m^* D_a F_m \frac{1}{m}F_m^* D_b F_m = \frac{1}{m}F_m^* D_a D_b F_m \text{ circulant.}
\end{aligned}
$$

These relations are shown in [2, p. 3].

1.4.2.2 Utilizing the circulant matrices
Using this background we now express our methods in the form $U^{n+1} = CU^n$ with C being a circulant matrix.

With forwards Euler this is straight forward as our system is in the correct form, with the matrix C defined by its first column vector $\mathbf{c} = [c_0, c_1, \ldots, c_m]^T$ where all values $c_i = 0$ except $c_0 = 1$, $c_1 = -c_{m-1} = -\frac{r}{8}(3 - 4h^2(1 + \pi^2))$ and $c_3 = -c_{m-3} = \frac{r}{8}$.

With Crank-Nicolson we have an equation on the form $AU^{n+1} = BU^n$ with A, B circulant. A simple rearrangement shows that

$$\begin{aligned} AU^{n+1} = BU^n \implies U^{n+1} =& A^{-1}BU^n \\ =& \left(\frac{1}{m}F_m^* D_a^{-1} F_m\right)\left(\frac{1}{m}F_m^* D_a F_m\right)U^n \\ =& \frac{1}{m}F_m^* D_a^{-1} D_b F_m U^n = CU^n \end{aligned}$$

with C a circular matrix. The calculation of $D_a^{-1}D_b$ is done in the code implementation, but it can be noted that the matrix A is defined by its first column-vector $\mathbf{a}$ with $a_0 = 1$, $a_1 = -a_{m-1} = \frac{r}{16}(3 - 4h^2(1 + \pi^2))$ and $a_3 = -a_{m-3} = -\frac{r}{16}$. Similarly B is defined by its first column vector $\mathbf{b}$ where $b_0 = 1$, $b_1 = -b_{m-1} = -\frac{r}{16}(3 - 4h^2(1 + \pi^2))$ and $b_3 = -b_{m-3} = \frac{r}{16}$.

Using this method we can calculate

$$\begin{aligned} U^n =& CU^{n-1} = C(CU^{n-2}) = \cdots = C^n U^0 \\ =& \left(\frac{1}{m}F_m^* D_c F_m\right)^n U^0 = \frac{1}{m}F_m^* D_c^n F_m U^0. \end{aligned}$$

which is far more efficient than the straight-forward method for large m and n, especially in the case of the CN-method where either a set of linear equations must be solved for each step or A^{-1} must be found.

1.4.2.3 Convergence plots Figure 14 shows that for a constant $k = t/N$ the scheme for the forwards Euler method never converges for larger $M = 1/(2h)$, in fact it diverges rapidly. This is related to the increase in $\mu = k/2h^6$ with higher M. The plot also shows that the Crank-Nicolson method converges for larger M with a rate that is $O(h^2)$, as is expected for a constant k since our approximation in the CN-scheme discards terms of order $O(kh^2 + k^3)$. The small increase in error at the end is then probably due to the term k^3 in our approximation, which is independent of h but we observe to decrease with higher N.

Contrasting Figure 14, Figure 15 shows that when keeping $\mu = k/2h^6$ constant the forwards Euler method converges for a small enough μ when increasing M. When μ is constant both the forwards Euler method and the Crank-Nicolson scheme converges with a rate $O(h)$.

1.4.3 Conservation of L_2 and ℓ_2 norms

We have the analytical solution to the given KdV-problem given by

$$\begin{aligned} u(x,t) &= \sum_{k\in\mathbb{Z}} \widehat{u}(k,0)\exp\left(-\mathrm{i}\pi k\left(1+\pi^2\right)t + \mathrm{i}\pi^3 k^3 t\right)\exp(\mathrm{i}\pi k x) \\ \text{where } \widehat{u}(k,t) &= \frac{1}{2}\int_{-1}^{1} u(x,t)\exp(-\mathrm{i}\pi k x)\mathrm{d}x. \end{aligned}$$

We then use the orthogonality of $\exp(-i\pi kx)$ on $x \in (-1, 1)$, and when using the common inner product induced by the given L_2 norm on functions, we have $||u(x,t)||_2^2 = \langle u, u\rangle$ (we are using the inner product over x, not t). Then all the cross-terms in the sum give 0 contribution to the inner product, and we are left with the sum of the inner products of the terms. We write

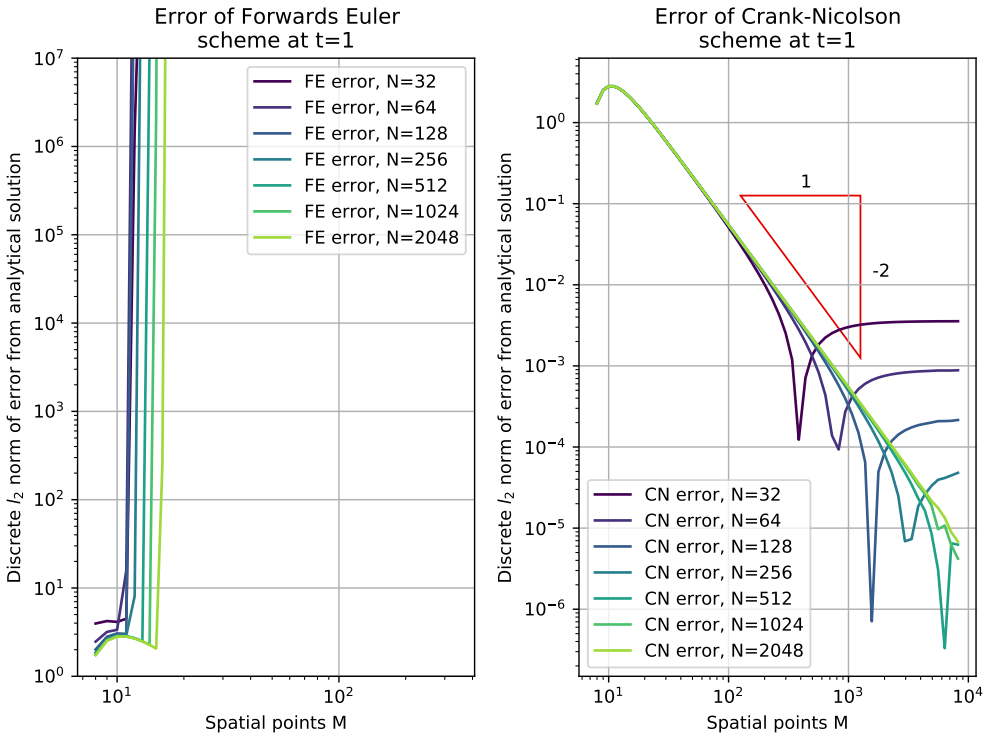


Figure 14: Convergence of errors in the given linearized KdV problem implementation. The plot is made by using the forwards Euler and Crank-Nicolson method respectively. As an initial condition $\sin(\pi x)$ was given, and the scheme was implemented at $x \in [-1, 1]$ with a periodic boundary condition. The analytical solution of this problem is $\sin(\pi(x - t))$. The plotted error is the difference between the exact analytical solution and the solution found by the numerical schemes in the discrete ℓ_2 norm.

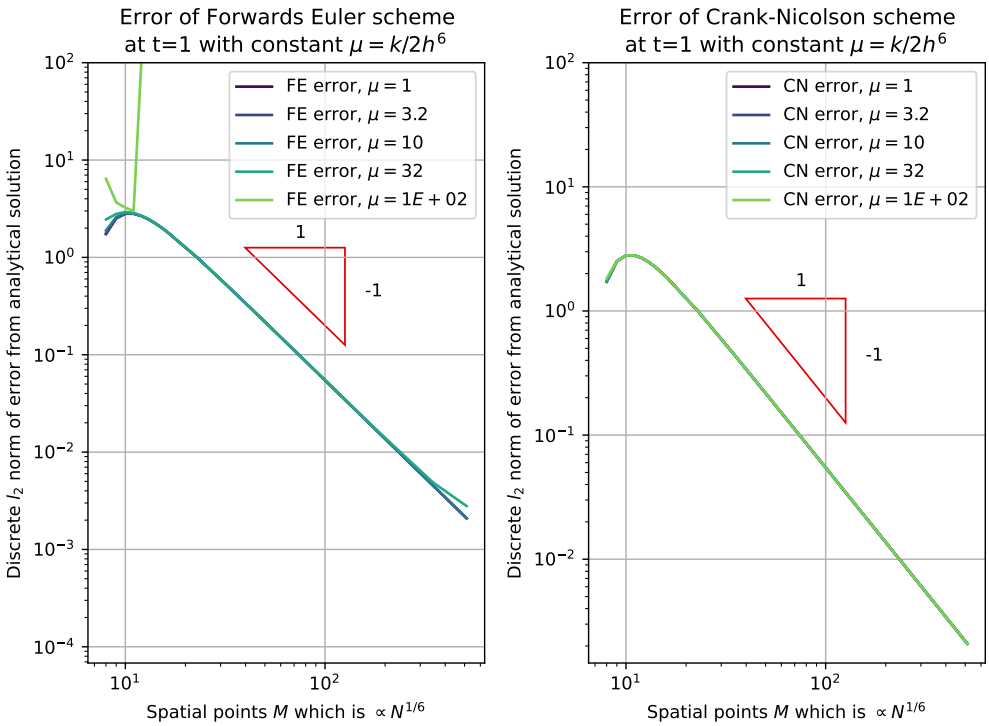


Figure 15: Convergence of errors for the given problem. In this convergence plot the spatial resolution $M \propto N^{1/6}$ with $\mu = k/2h^6$. As a consequence the Euler method is von Neumann stable, and will converge for small enough μ. We observe that for small enough μ the method converges even for the forwards Euler implementation. For further problem description, see of 14.

$$
\begin{aligned}
||u(x,t)||_2^2 = \langle u(x,t), u\rangle \quad \text{where } u(x,t) = & \sum_{k\in\mathbb{Z}} \widehat{u}(k,0)\exp\left(-\mathrm{i}\pi k\left(1+\pi^2\right)t + \mathrm{i}\pi^3k^3t\right)\exp(\mathrm{i}\pi kx)\\
= \sum_{k\in\mathbb{Z}} \langle v_k, v_k\rangle \quad \text{where } v_k(x,t) = & \quad \widehat{u}(k,0)\exp\left(-\mathrm{i}\pi k\left(1+\pi^2\right)t + \mathrm{i}\pi^3k^3t\right)\exp(\mathrm{i}\pi kx)
\end{aligned}
$$

which gives

$$
\begin{aligned}
\langle v_k, v_k\rangle =& \frac{1}{2}\int_{-1}^{1} v_k(x,t)\overline{v_k(x,t)}\mathrm{d}x\\
=& \frac{1}{2}\int_{-1}^{1} \widehat{u}(k,0)^2\exp\left(-\mathrm{i}\pi k\left(1+\pi^2\right)t + \mathrm{i}\pi^3k^3t\right)\overline{\exp\left(-\mathrm{i}\pi k\left(1+\pi^2\right)t + \mathrm{i}\pi^3k^3t\right)}\\
&\cdot \exp(\mathrm{i}\pi kx)\overline{\exp(\mathrm{i}\pi kx)}\mathrm{d}x\\
=& \frac{1}{2}\int_{-1}^{1} \widehat{u}(k,0)^2\exp([-\mathrm{i}\pi k\left(1+\pi^2\right)t + \mathrm{i}\pi^3k^3] - [-\mathrm{i}\pi k\left(1+\pi^2\right)t + \mathrm{i}\pi^3k^3t])\\
&\cdot \exp(\mathrm{i}\pi kx - \mathrm{i}\pi kx)\mathrm{d}x\\
=& \frac{1}{2}\int_{-1}^{1} \widehat{u}(k,0)^2\cdot 1\cdot 1\cdot \mathrm{d}x = \frac{1}{2}\int_{-1}^{1}\widehat{u}(k,0)^2 = C_k\\
\implies ||u(x,t)||_2^2 =& \sum_{k\in F}\langle v_k, v_k\rangle = \sum_{k\in F} C_k
\end{aligned}
$$

Which is constant over t seen by its independence of the time parameter.

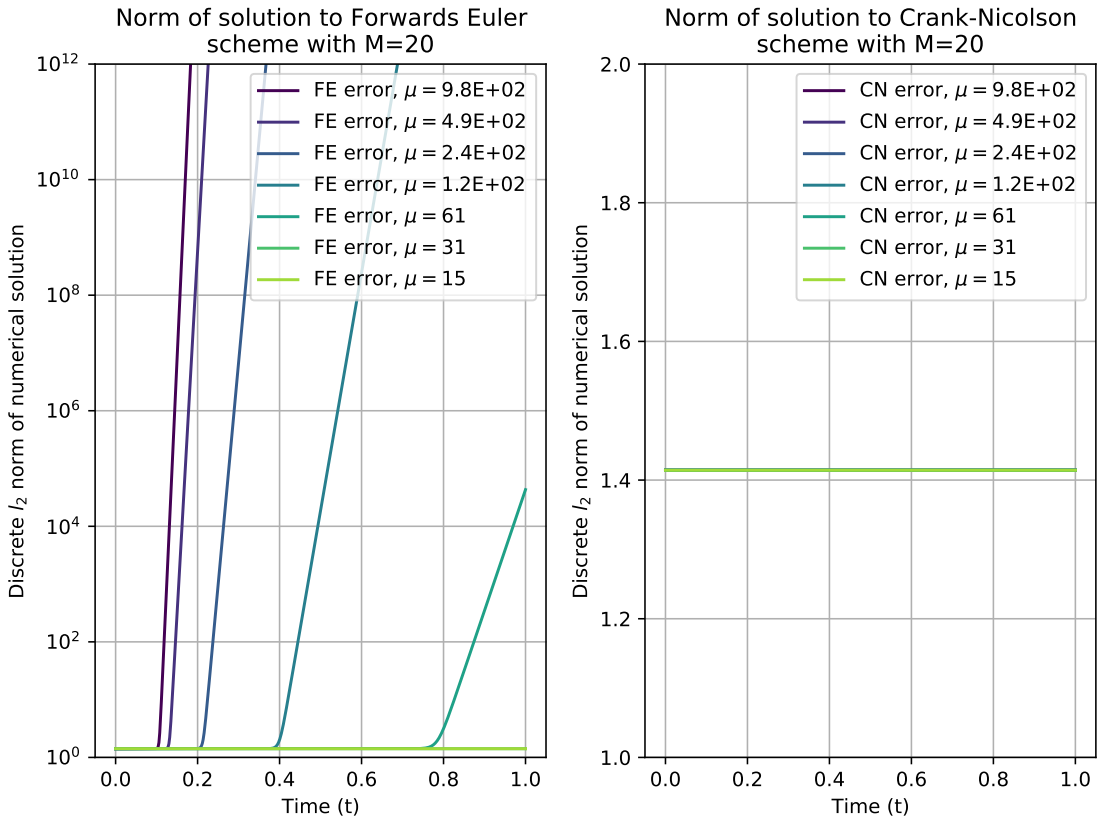


Figure 16: Discrete ℓ_2 norm of numerical solution to the KdV equation with initial condition $u(x,0) = \sin(\pi x)$ on $x \in (-1,1)$. The plot shows the divergence of the forwards-Euler method after a few time steps for large μ, and convergence towards the correct norm for smaller μ. Additionally we see the constant norm of the CN solution to the KdV problem for the given initial condition. The largest difference in ℓ_2 norm from $t = 0$ to $t = 1$ is of the order 10^{-12} for the CN method (not visible in the graph), probably due to numerical imprecision.

As seen from the plots in figure 16 using the CN-procedure, this analytical result holds (up to numerical precision) for the stable CN-algorithm as well as for the FE method with small enough μ. Again we observe the convergence of the forwards Euler method for small μ as in the previous figures.

1.5 Finite element method for Poisson equation

This section will once again focus on the one-dimensional Poisson equation, although in the setting of finite element methods and adaptive finite element methods. We also consider the inhomogeneous pure Dirichlet boundary condition, i.e.

$$-u_{xx} = f(x) \quad u(a) = d_1, u(b) = d_2 \tag{39}$$

On the domain $[0, N]$, we consider the grid $\left\{a + n\frac{b-a}{N}\right\}_{n=0}^{N}$, i.e. a uniform grid between $[a, b]$.

We first derive the mathematical foundations for finite element method (FEM) and adaptive finite element method (AFEM). Thereafter, we describe and highlight finds from the implementation of said methods.

1.5.1 Mathematical formulation

1.5.1.1 Weak formulation

We can relate the solution of our inhomogeneous Dirichlet problem to the solution of the homogeneous problem. First we find the weak form of the homogeneous problem

$$-\int_a^b u_{xx}(x)v(x)dx = \int_a^b f(x)v(x)dx, \tag{40}$$

where the test function v obey $v(a) = v(b) = 0$. Consider the left hand side of (40). Using integration by parts, we get

$$\begin{aligned} -\int_a^b u_{xx}(x)v(x)dx &= \int_a^b u_x(x)v_x(x)dx \qquad\qquad - [u_x(x)v(x)]_a^b \\ &= \int_a^b u_x(x)v_x(x)dx \end{aligned}$$

assuming all $v \in H^1$ satisfy (39).

Thus, we get the weak formulation

$$\int_a^b u_x(x)v_x(x)dx = \int_a^b f(x)v(x)dx, \tag{41}$$

of the homogeneous problem. To solve the inhomogeneous problem, we solve the homogeneous first, and apply the boundary conditions afterwards.

We are considering a linear finite element method, so we are looking for solutions that lie in the space of piecewise linear functions. Let X_h^1 denote the space of all these piecewise linear functions. A basis for X_h^1 is expressed by the basis functions $\varphi_i(x)$, i.e. the elements of X_h^1 that evaluate to 1 at the points x_i and zero at all other points x_j. The basis functions are given in [3, p. 6] by

$$\begin{aligned} \varphi_0(x) &= \frac{x_1 - x}{x_1 - x_0} \quad , x_0 \le x \le x_1 \\ \varphi_i(x) &= \begin{cases} \frac{x - x_{i-1}}{x_i - x_{i-1}} & , x_{i-1} \le x \le x_i \\ \frac{x_{i+1} - x}{x_{i+1} - x_i} & , x_i \le x \le x_{i+1} \end{cases} \\ \varphi_N(x) &= \frac{x - x_N}{x_N - x_{N-1}} \quad , x_{N-1} \le x \le x_N. \end{aligned}$$

1.5.1.2 Setting up a linear system

By expanding the weak formulation (41) in terms of the basis, we can set up a linear system whose solution is the numerical solution of (39). Since we have an inhomogeneous Dirichlet problem, suppose we can write our vector with solutions along the x-axis as

$$\mathbf{u} = \hat{\mathbf{u}} + \tilde{\mathbf{u}}, \ \tilde{\mathbf{u}} = (d_1, 0, ..., 0, d_2),$$

where $\hat{u}$ is the homogeneous solution. Thus, our linear system can be written as

$$A\mathbf{u} = \mathbf{f} \ \Rightarrow \ A\hat{\mathbf{u}} = \mathbf{F} - A(d_1, 0, \ldots, 0, d_2), \tag{42}$$

where $\mathbf{F}$ is the loading vector for the homogeneous problem. What remains is to find expressions for A and for $\mathbf{F}$. A is found by initializing an $N \times N$ matrix with only zeros. Then we add

$$\frac{1}{h}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

to the submatrix $[A_{ij}]_{i,j=k,k+1}$. For instance, with uniform grid (i.e. without AFEM), A becomes

$$A = \frac{1}{h}\begin{bmatrix} 1 & -1 & 0 & \dots & \dots & 0 \\ -1 & 2 & -1 & \ddots & & 0 \\ 0 & -1 & 2 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & -1 & 2 & -1 \\ 0 & \dots & \dots & 0 & -1 & 1 \end{bmatrix},$$

where $h = x_{i+1} - x_i$.

We can find F by elementwise integrating

$$F_i = \int_{x_i}^{x_i+1} f(x)\varphi_i(x)dx,$$

which can be calculated using Gaussian quadrature. Now all we have to do is solve (42), which we can do by first solving for $\hat{\boldsymbol{u}}$, where the vector $\mathbf{f}$ on the right hand side is calculated first by matrix multiplication before we remove the first and last elements, which corresponds to enforcing the homogeneous boundary conditions. And so we can find the homogeneous solution $\hat{\mathbf{u}}$. We expand $\hat{\mathbf{u}}$ with d_1 as its first element and d_2 as its last element, thus imposing the inhomogenous boundary conditions. Finally, we may find the solution

$$u_h(x) = \sum_i \hat{u}(x_i)\varphi_i(x_i)$$

.

1.5.1.3 Analytical solutions to different parameters

We use FEM on a set of test equations whose analytical solution exists as described in Table 1.

Integrating $-f(x)$ twice as in section 1.1.1.1 yields

b) $x^2 + c_1x + c_2$

c) $\exp(-100x^2) + c_3x + c_4$

d) $\exp(-1000x^2) + c_5x + c_6$

e) $x^{\frac{2}{3}} + c_7x + c_8$

Imposing the boundary conditions (d_1, d_2) as given in Table 1, it is clear that all coefficients $c_1 = c_2 = \cdots = c_8 = 0$, and so we get the analytical solutions

b) x^2

c) $\exp(-100x^2)$

d) $\exp(-1000x^2)$

e) $x^{\frac{2}{3}}$

Table 1: Overview of differing parameters for the equations solved with FEM.

Case	$f(x)$	(d_1, d_2)	(a, b)
b)	-2	$(0, 1)$	$(0, 1)$
c)	$-(4 \cdot 10^4 x^2 - 200)\exp(-100x^2)$	$(\exp(-100), \exp(-100))$	$(-1, 1)$
d)	$-(4 \cdot 10^6 x^2 - 2000)\exp(-1000x^2)$	$(\exp(-1000), \exp(-1000))$	$(-1, 1)$
e)	$\frac{2}{9}x^{-\frac{4}{3}}$	$(0, 1)$	$(0, 1)$

1.5.2 Computational aspects

The more computationally difficult aspect when solving PDEs with FEM, is the fact that the load vector ($\mathbf{F}$) often has to be integrated using Gaussian quadrature. In principle, Gaussian quadrature is based on expressing an integral $I_w(f) = \int_a^b w(x)f(x)dx, \quad w(x) \geq 0$, as a sum

$$I_{n,w}(f) = \sum_{i=1}^{n} \alpha_i f(x_i),$$

where $w(x)$ is a weight function and α_i are coefficients to be determined [1]. Luckily, there are methods within the NumPy and SciPy modules automatically determines the weights and coefficients, automating the Gaussian quadrature process. In algorithm 3 we describe the principles of implementing finite element method for the one-dimensional Poisson equation.

Similarly to the refinement techniques using finite difference schemes, we can implement an adaptive finite element method with non-uniform grids. We determine if

$$\|U_h(x_i) - u(x_i)\|_{L_2} > \alpha E, \tag{43}$$

where E is either

$$\max_i \|U_h(x_i) - u(x_i)\|_{L_2} \quad \text{with } \alpha = 1, \text{ or} \tag{44}$$

$$\frac{\|U_h(x) - u(x)\|_{L_2}}{N} \quad \text{with } \alpha = 0.7, \tag{45}$$

where the norm in (45) is over its entire domain. (44) is referred to as the max-strategy, and (45) is referred to as the averaging-strategy.

In contrast to refining finite difference schemes, we do not need to interpolate the approximations, since the result from FEM itself is a continuous function of x. The implementation of adaptive FEM is described in Algorithm 4.

1.5.3 Numerical examples

The analytical solution, $u(x)$, is compared to the numerical solution with FEM in Figure 17 for case b and c, and 18 for case d and e. After the first step, where $N = 8$, FEM manages to get the contours of the function, especially for case b and e. Already at the forth step where $N = 64$, we observe almost indistinguishable values when examining at the entire domain. The exceptions are the exponential functions, c and d, in which the error is a little larger at the peak of the function.

Furthermore, Figure 19 shows the relative L_2-norm error for case b, c, d and e with parameters from Table 1, as a result of FEM with $N = 2^3, 2^4, \ldots, 2^{11}$ elements. In addition, we plot the relative L_2-norm error from adaptive FEM, both with the averaging strategy and max strategy as described in Algorithm 4.

The error plots confirm the visual error observed in Figures 17 and 18, seeing as the relative error is $\sim 10^{-3}$ for 19b and 19e for the forth node, whereas the relative errors in 19c and 19d are over 10^{-2} for the same node. However, all cases except for e seem to converge with $\mathcal{O}(h^2)$ in error with increasing Ndofs.

Algorithm 3 Finite element method for (39).

procedure FEM
  input: N number of elements, parameters from Table 1, basis function $\varphi_i(x)$
  Create uniform grid X with elements x_n, s.t. for $x_n = \{a + n\frac{b-a}{N}\}_{n=0}^{N}$
  Initialize $A \in \mathbb{R}^{N\times N}$ and $F \in \mathbb{R}^N$
  for i=1, ... N-2 **do**
    Add the 2×2 matrix

$$\frac{1}{x_{i+1} - x_i}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$$

    to the submatrix $[A_{kl}]_{k,l=i,i+1}$
    Add to F, using Gaussian quadrature,

$$\int_{x_i}^{x_{i+1}} f(x)\varphi_{i+1}(x)dx$$

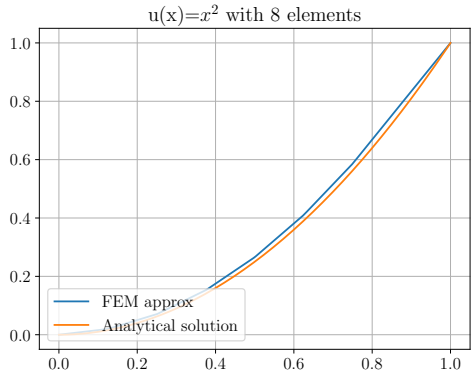
  Add also to the last index of F:

$$\int_{x_N-1}^{x_N} f(x)\varphi_N(x)dx$$

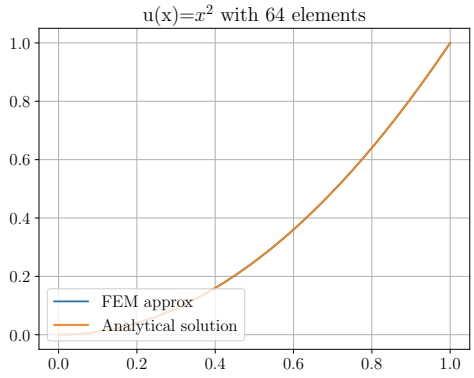
  Let $\tilde{u} \in \mathbb{R}^N$ be defined by $\tilde{u} := (d_1, 0, \ldots, 0, d_2)$
  $LHS \leftarrow A$ with the first and last rows and columns removed
  $RHS \leftarrow F - A\tilde{u}$ with the first and last elements removed
  Solve $LHS\hat{u} = RHS$ for $\hat{u}$
  Expand dimensions of $\hat{u}$ by 2, inserting d_1 to the left and d_2 to the right of the vector.
  Transform $\hat{u}$ to a functional $u_h(x) := \sum_i \hat{u}_i\varphi_i(x)$
  return $u_h(x)$

Algorithm 4 Adaptive refinement with finite element method. Refines grid X based on refinement criteria

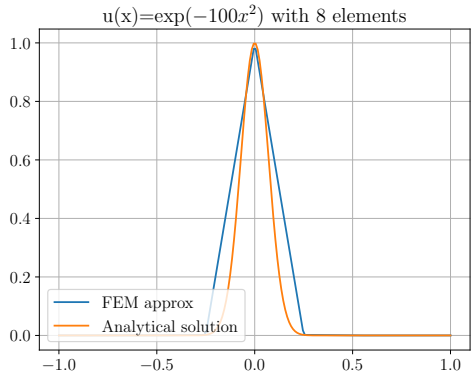
procedure AFEM
  input: Grid X, refinement method, BVP-problem with analytical solution $u(x_i)$ and approximated solution $U(x_i)$.
  for all $x_i \in X$ **do**
    Determine E based on (44) or (45)
    if $\|U_h(x_i) - u(x_i)\|_{L_2} > \alpha E$ **then**
      Find midpoint $\frac{1}{2}(x_i + x_{i+1})$ and store in some vector X_{ref}
  Update $X \leftarrow X + X_{ref}$ and sort grid
  Update H s.t. $h_i = x_{i+1} - x_i \forall h_i \in H$
  Find a new $u_h(x)$ with the refined mesh using Algorithm 3
  return Approximated solution using FEM on refined, non-uniform grid



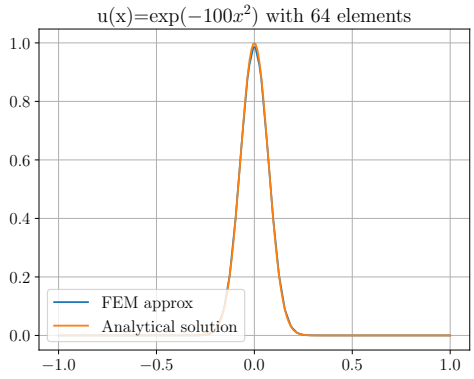
Case b, first step, $N = 8$



case b, forth step, $N = 64$



Case c, first step, $N = 8$



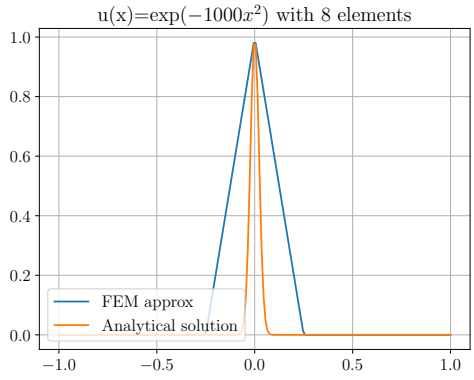
Case c, forth step, $N = 64$

Figure 17: Plots of $u(x)$ on the domain $[a, b]$ for case b and c. We plot the first and forth step, and observe that the approximation and analytical solution are indistinguishable already at the forth step, with the exception of some error at the peak for case c.

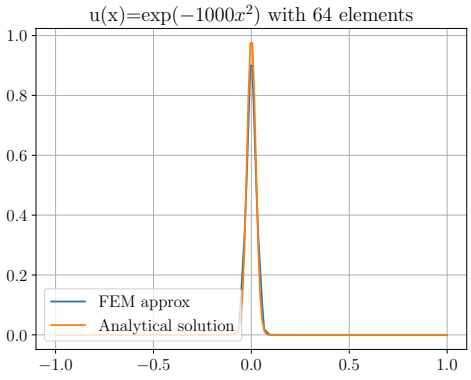
Generally, the errors using AFEM seem to be relatively similar when comparing the max strategy and averaging strategy with the chosen α.

For 19b, the adaptive refinement does not seem to help convergence rate compared to uniform refinement. The exponential cases, 19c and 19d, show a great decrease in error using AFEM, and 19e shows some decrease, although not as dramatic as the two other. Considering the geometrical properties of the exponential functions, it makes sense that more dense points around the spike constitutes a better approximation.

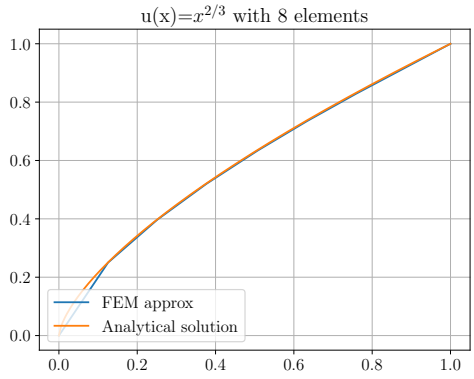
One other noteworthy property from Figure 18, is the convergence rate of 19e being considerably worse than the other convergence rates. The high inaccuracy is explained upon examining $u(x) = x^{2/3}$ and $u'(x)$ when $x \to 0$. Indeed $u(x)$ is not smooth in $x = 0$ as $u'(x) \to \infty$.



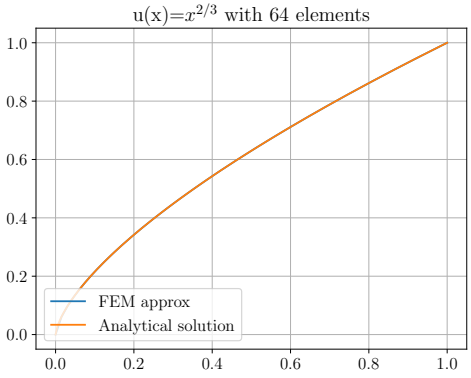
Case d, first step, $N = 8$



Case d, forth step, $N = 64$



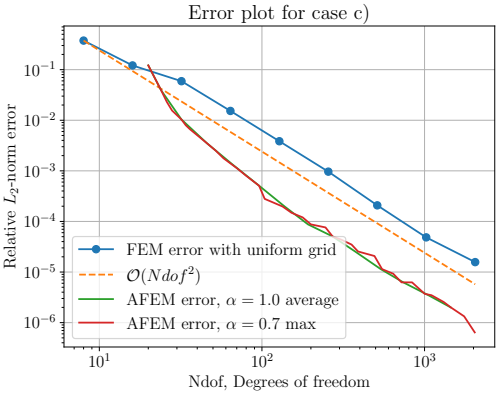
Case e, first step, $N = 8$



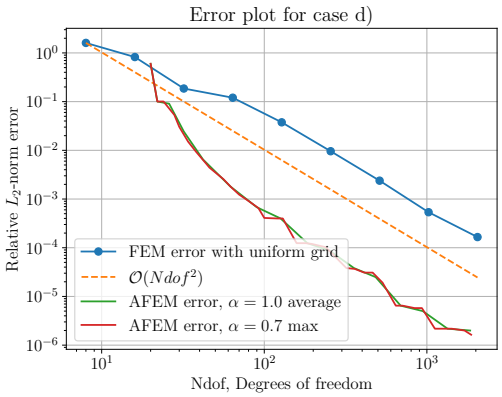
Case e, forth step, $N = 64$

Figure 18: Plots of $u(x)$ on the domain $[a, b]$ for case d and e. We plot the first and forth step, and observe that the approximation and analytical solution are almost indistinguishable for case e. However, the error is greater at the peak of case d.

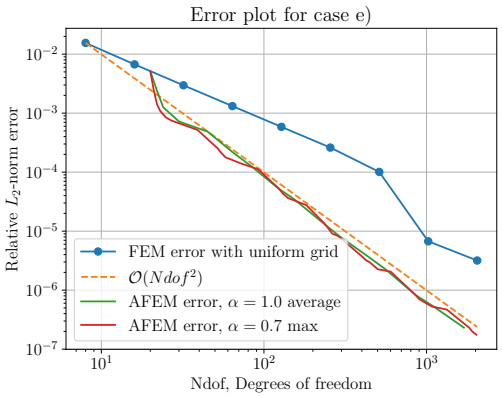
(b) $u(x) = x^2$



(c) $u(x) = \exp(-100x^2)$



(d) $u(x) = \exp(-1000x^2)$



(e) $u(x) = x^{\frac{2}{3}}$

Figure 19: Error plots for FEM on case b through e.

2 The Biharmonic Equation

2.1 Solution as a Fourier series

In this part we consider the inhomogenous Biharmonic equation with clamped boundary conditions on $\Omega = [0,1]^2$,

$$\begin{cases} \nabla^4 u = f & (x,y) \in \Omega \\ u = 0,\ \nabla^2 u = 0 & (x,y) \in \partial\Omega \end{cases} \tag{46}$$

Assume $u, f \in L^2(\Omega)$. For any fixed $y_0 \in [0,1]$, we may expand $u(x,y_0)$ and $f(x,y_0)$ in Fourier series in x,

$$u(x,y_0) = \sum_{m=1}^{\infty} F_m(y_0) \sin m\pi x,$$
$$f(x,y_0) = \sum_{m=1}^{\infty} f_m(y_0) \sin m\pi x,$$

where F_m, f_m are functions of y_0, but constants with respect to x. We may expand F_m, f_m in the same way with respect to y_0, so that exchanging y_0 for y yields

$$u(x,y) = \sum_{m=1}^{\infty}\sum_{n=1}^{\infty} F_{mn} \sin m\pi x \sin n\pi y, \tag{47}$$

$$f(x,y) = \sum_{m=1}^{\infty}\sum_{n=1}^{\infty} f_{mn} \sin m\pi x \sin n\pi y. \tag{48}$$

Define $K_{mn}(x,y) := \sin(m\pi x)\sin(n\pi y)$ and note that

$$K_{mn} = 0 \quad \text{on } \partial\Omega$$

and

$$\nabla^2 K_{mn} = -(m^2+n^2)\pi^2 K_{mn} = 0 \quad \text{on } \partial\Omega,$$

so u satisfies the boundary conditions in (46) regardless of the constant F_{mn}. Also,

$$\nabla^4 K_{mn} = (m^4 + 2m^2n^2 + n^4)\pi^4 K_{mn},$$

that is, K_{mn} is an eigenfunction to the ∇^4 operator with eigenvalue $(m^4+2m^2n^2+n^4)\pi^4$. Thus we know from (46) that

$$\sum_{m=1}^{\infty}\sum_{n=1}^{\infty} F_{mn}(m^4+2m^2n^2+n^4)\pi^4 K_{mn} = \sum_{m=1}^{\infty}\sum_{n=1}^{\infty} f_{mn} K_{mn}. \tag{49}$$

Euler's orthogonality relations states that

$$\int_0^1 \sin(m\pi\xi)\sin(n\pi\xi)\mathrm{d}\xi = \begin{cases} 1, & \text{if } m = n \\ 0, & \text{if } m \neq n \end{cases}$$

Thus, multiplying both sides of (49) by $\sin(n_0\pi y)$ and integrating over y yields

$$\sum_{m=1}^{\infty} F_{mn_0}(m^4 + 2m^2n_0^2 + n_0^4)\pi^4 \sin(m\pi x) = \sum_{m=1}^{\infty} f_{mn_0}\sin(m\pi x). \tag{50}$$

The same procedure can be used to eliminate x and the sum over m, which in the end yields

$$\begin{aligned} F_{mn}(m^4 + 2m^2n^2 + n^4)\pi^4 &= f_{mn} \\ \implies F_{mn} &= \frac{f_{mn}}{(m^4 + 2m^2n^2 + n^4)\pi^4}. \end{aligned} \tag{51}$$

Thus, any choice of f such that only a single f_{nm} is nonzero collapses (47) into a single expression. This means any f of the form

$$f(x, y) = A\sin(m\pi x)\sin(n\pi y)$$

with A constant, produces a solution

$$u(x, y) = \frac{A\sin(m\pi x)\sin(n\pi y)}{(m^4 + 2m^2n^2 + n^4)\pi^4}.$$

2.2 System of Poisson equations

Equation (46) can be transformed into a system of inhomogenous Poisson equations by introducing $v = \nabla^2 u$. This yields

$$\begin{cases} \nabla^2 v = f, & v = 0 \text{ on } \partial\Omega \\ \nabla^2 u = v, & u = 0 \text{ on } \partial\Omega \end{cases} \tag{52}$$

2.3 System matrix for the 5 and 9 point stencil

In this section we consider only the solution of the system $\nabla^2 v = f$ on $\Omega = [0, 1]^2$ with $v = 0$ on $\partial\Omega$, as we have reduced the biharmonic problem to solving that system twice shown in the previous section.

In order to write our system matrices compactly we define the $M \times M$ Toeplitz, Symmetric and Tridiagonal (TST) matrix $T_M^{(\gamma)}$ as

$$T_M^{(\gamma)} = \begin{bmatrix} \gamma & 1 & & & \\ 1 & \gamma & 1 & & \\ & 1 & \gamma & \ddots & \\ & & \ddots & \ddots & 1 \\ & & & 1 & \gamma \end{bmatrix} \tag{53}$$

and note that the well known eigenvalues of TST-matrices ([6, p. 2]) give that the eigenvalues and corresponding eigenvectors of $T_M^{(\gamma)}$ are

$$\begin{aligned} \lambda_m &= \gamma + 2\cos\frac{m\pi}{M+1}, \qquad m = 1, \ldots, M \\ \boldsymbol{e}_m &= [\sin\omega_m, \sin 2\omega_m, \ldots, \sin M\omega_m]^T \quad \text{with } \omega_m = \frac{m\pi}{M+1} \end{aligned} \tag{54}$$

Furthermore, we note that in our system of equations we discretize with $i = 0, \ldots, N$ indexing along the x-axis and $j = 0, \ldots, N$ indexing along the y-axis. (We use the same spatial resolution along both axes for simplicity.) Using a uniform mesh on our region $\Omega \in [0,1]^2$ we then have $v_{(i,j)} = v(i/(N+1), j/(N+1))$. Let V be our numerical approximation to the exact solution v on the interior of Ω. We define V as the vector where the point $(i/(N+1), j/(N+1))$ is approximated by the entry $V_{(i,j)} = V_{i+(N-1)j}$.

We may then write, $TV = \mathbf{g}$ where T is an $(N-1)^2 \times (N-1)^2$ matrix and $\mathbf{g}$ is a vector of length $(N-1)^2$, analogous to $\nabla^2 v = f$. The matrix T and vector $\mathbf{g}$ will depend on our numerical differentiation scheme.

Note also that $\otimes$ is used as the Kronecker product.

2.3.1 Five point stencil

Using the five point stencil defined by

$$u_{i-1,j} + u_{i+1,j} - 4u_{ij} + u_{i,j-1} + u_{i,j+1} = h^2 f_{ij} \tag{55}$$

we get a matrix T and vector $\mathbf{g}$ where

$$T_{a,b} = \begin{cases} -4 & a = b \\ +1 & a = b \pm 1 \\ +1 & a = b \pm (N-1) \\ 0 & \text{otherwise} \end{cases} \qquad \text{for } a, b \in \{1, ..., N-1\}$$

$$g_{i+(N-1)j} = g_{(i,j)} = f_{(i,j)} = f\left(\frac{i}{(N+1)}, \frac{j}{(N+1)}\right)$$

Using our previously defined $T_n^{(\gamma)}$ we may then write

$$T = \begin{bmatrix} T_{N-1}^{(-4)} & I_{N-1} & & & \\ I_{N-1} & T_{N-1}^{(-4)} & I_{N-1} & & \\ & I_{N-1} & T_{N-1}^{(-4)} & \ddots & \\ & & \ddots & \ddots & I_{N-1} \\ & & & I_{N-1} & T_{N-1}^{(-4)} \end{bmatrix} = T' \otimes I + I \otimes T' \tag{56}$$

where $T' = T_{N-1}^{(-2)}$ and $I = I_{N-1}$.

We can easily find the eigenvalues of each of the blocks in this matrix, as well as the components of the Kronecker product sum, using the formulas given in 2.c. For the identity matrix I, the eigenvalues are 1 (and the set of eigenvectors the whole $\mathbb{C}^{N-1}$). For $T^{(-4)}$ we have eigenvalues and -vectors given by equation (54) which in this case gives $\lambda_n = -4 + 2\cos(n\pi/N)$ for $n \in 1, N-1$. For T' we have $\lambda_n = -2 + 2\cos(n\pi/N)$ for $n \in 1, N-1$.

2.3.2 Nine point stencil

Using the nine point stencil defined by

$$
\begin{aligned}
&\frac{1}{6}\left(u_{i+1,j+1}+u_{i-1,j+1}+u_{i+1,j-1}+u_{i-1,j-1}\right)\\
+&\frac{2}{3}\left(u_{i+1,j}+u_{i-1,j}+u_{i,j+1}+u_{i,j-1}\right)-\frac{10}{3}u_{ij}\\
=&h^2\left[\frac{2}{3}f_{ij}+\frac{1}{12}\left(f_{i+1,j}+f_{i-1,j}+f_{i,j+1}+f_{i,j-1}\right)\right]
\end{aligned}
\tag{57}
$$

we get a matrix T and vector $\mathbf{g}$ where

$$
T_{a,b}=\begin{cases}
-\frac{10}{3} & a=b\\
+\frac{2}{3} & a=b\pm 1\\
+\frac{2}{3} & a=b\pm(N-1)\\
+\frac{1}{6} & a=b\pm 1\pm(N-1)\\
0 & \text{otherwise}
\end{cases}
\qquad \text{for } a,b\in\{1,...,N-1\}
\tag{58}
$$

$$
g_{i+(N-1)j}=g_{(i,j)}=h^2\left[\frac{2}{3}f_{ij}+\frac{1}{12}\left(f_{i+1,j}+f_{i-1,j}+f_{i,j+1}+f_{i,j-1}\right)\right]
\tag{59}
$$

Using our previously defined $T_n^{(\gamma)}$ we may then write

$$
T=\frac{1}{6}\begin{bmatrix}
4T_{N-1}^{(-5)} & T_{N-1}^{(4)} & & & \\
T_{N-1}^{(4)} & 4T_{N-1}^{(-5)} & T_{N-1}^{(4)} & & \\
 & T_{N-1}^{(4)} & 4T_{N-1}^{(-5)} & \ddots & \\
 & & \ddots & \ddots & T_{N-1}^{(4)}\\
 & & & T_{N-1}^{(4)} & 4T_{N-1}^{(-5)}
\end{bmatrix}=\frac{1}{6}(T'\otimes T')+T'\otimes I+I\otimes T'.
\tag{60}
$$

where $T'=T_{N-1}^{(-2)}$.

Again, we can easily find the eigenvalues of each of the blocks in this matrix. For $T^{(-5)}$, $T^{(4)}$ and T' we have eigenvalues and -vectors given by equation (54).

Multiplying by the factors 1/6 and 4/6 for the first two respectively, we find that for each block we have eigenvalues

$$
\begin{aligned}
\frac{4}{6}T^{(-5)}:&\quad \lambda_n=-\frac{10}{3}+\frac{4}{3}\cos(n\pi/N) &&\text{for } n\in 1,N-1.\\
\frac{1}{6}T^{(4)}:&\quad \lambda_n=\frac{2}{3}+\frac{1}{3}\cos(n\pi/N) &&\text{for } n\in 1,N-1\\
\text{and } T':&\quad \lambda_n=-2+2\cos(n\pi/N) &&\text{for } n\in 1,N-1.
\end{aligned}
$$

2.4 Convergence of 5 point stencil

The 5 point stencil $\nabla_5^2 u_p$ is stable and of order 2, which we aim to prove in this section. In order to prove this claim, we must first show that the difference equation in the stencil satisfies the criteria for the discrete maximum principle. The difference equation corresponding to $-\nabla_5^2 u_p$ is

$$
4u_p-(u_e+u_w+u_n+u_s).
\tag{61}
$$

Comparing this to the general discretization difference equation

$$
a_0u_p-\sum_{i=1}^{s}a_iu_{Q_i},
\tag{62}
$$

we see that all coefficients in (61) are positive, and the coefficient matrix is diagonally dominant. Thus the discrete maximum principle holds for $-\nabla_5^2$ with any discrete function v defined on all points $p \in \Omega \cup \partial\Omega$ such that

$$4v_p - (v_e + v_w + v_n + v_s) \leq 0$$

for all $p \in \Omega$. This is the same as requiring

$$-4v_p + v_e + v_w + v_n + v_s \geq 0, \tag{63}$$

so any v satisfying this will also attain its maximum value on $\partial\Omega$. Since (63) is the difference equation corresponding to ∇_5^2, any v satisfying $\nabla_5^2 v_p \geq 0$ for all $p \in \Omega$ will attain its maximum on $\partial\Omega$.

We now use this result to prove that if v is some discrete function defined on $\Omega \cup \partial\Omega$ such that $v_p = 0$ for any $p \in \partial\Omega$, then

$$\|v\|_\infty \leq \frac{1}{8} \left\|\nabla_5^2 v\right\|_\infty .$$

Let v be a discrete function that equals 0 on the boundary. The notation $\nabla_5^2 v$ will denote the 5 point stencil matrix (56) applied on v, while $\nabla_5^2 v_p$ still denotes the 5 point stencil (55) applied to v_p. Define f as the vector of all $f_p := \nabla_5^2 v_p$. Then

$$-\|f\|_\infty \leq \nabla_5^2 v_p \leq \|f\|_\infty . \tag{64}$$

As in [8], we define the function

$$w(x,y) := \frac{1}{4}\left[(x-1/2)^2 + (y-1/2)^2\right]. \tag{65}$$

Since

$$\begin{aligned} w(x \pm h, y) &= w(x,y) + \frac{1}{4}[h^2 \pm (2x-1)h], \\ w(x, y \pm h) &= w(x,y) + \frac{1}{4}[h^2 \pm (2y-1)h], \end{aligned}$$

we get

$$\begin{aligned} \nabla_5^2 w(x,y) &= \frac{1}{h^2}\left[w(x+h,y) + w(x-h,y) + w(x,y+h) + w(x,y-h) - 4w(x,y)\right] \\ &= \frac{1}{h^2}\left[4w(x,y) + h^2 - 4w(x,y)\right] \\ &= 1. \end{aligned}$$

Note also that w is a non-negative function. Define now $V_p^- := \|f\|_\infty w_p - v_p$. By (64),

$$\nabla_5^2 V_p^- = \|f\|_\infty - f_p \geq 0,$$

so the discrete maximum principle holds for the discrete function V^-. That is,

$$\begin{aligned} -v_p \leq \|f\|_\infty w_p - v_p &\leq \max_{p\in\partial\Omega}[\|f\|_\infty w_p - v_p] \\ &= \max_{p\in\partial\Omega}[\|f\|_\infty w_p] \\ &= \frac{1}{8}\|f\|_\infty , \end{aligned} \tag{66}$$

since $\max_{p\in\partial\Omega} w_p = 1/8$ and $v_p = 0$ on $\partial\Omega$. Defining now $V_p^+ := \|f\|_\infty w_p + v_p$, we still have

$$\nabla_5^2 V_p^+ = \|f\|_\infty + f_p \geq 0,$$

so the discrete maximum principle holds for V^+. This means

$$\begin{aligned} v_p \leq \|f\|_\infty w_p + v_p &\leq \max_{p\in\partial\Omega}[\|f\|_\infty w_p + v_p] \\ &= \frac{1}{8}\|f\|_\infty . \end{aligned} \tag{67}$$

Putting (66) and (67) together, we get

$$|v_p| \leq \frac{1}{8} \|f\|_\infty ,$$

and thus,

$$\|v\|_\infty \leq \frac{1}{8} \left\|\nabla_5^2 v\right\|_\infty . \tag{68}$$

Finally, we use this to bound the global error. Define u, v by $\nabla^2 u = f$, $\nabla_5^2 v = f$. The truncation error at p is

$$\tau_p = \begin{cases} \nabla_5^2 u_p - f_p, & p \in \Omega \\ 0, & p \in \partial\Omega \end{cases} \tag{69}$$

On a uniform orthogonal grid with spacing h,

$$\delta_\xi^2 u_p = \partial_\xi^2 u_p + \frac{h^2}{12} \partial_\xi^4 u_p + \mathcal{O}(h^4), \quad \xi \in \{x, y\}$$

so for small enough h,

$$\begin{aligned} |\tau_p| &= |\delta_x^2 u_p + \delta_y^2 u_p - f_p| \\ &= |\nabla^2 u_p - f_p + \frac{h^2}{12} [\partial_x^4 + \partial_y^4] u_p + \mathcal{O}(h^4)| \\ &= \frac{h^2}{12} K \end{aligned} \tag{70}$$

where $K = \max_{p \in \Omega} \{|[\partial_x^4 + \partial_y^4] u_p\}$. Define $\bar{\tau} := \frac{h^2}{6} K$ and $e_p := u_p - v_p$. It is easy to see that

$$\nabla_5^2 e_p = \nabla_5^2 u_p - \nabla_5^2 v_p = \tau_p, \quad p \in \Omega \tag{71}$$

$$\nabla_5^2 e_p = 0, \quad p \in \partial\Omega \tag{72}$$

Thus,

$$|\nabla_5^2 e_p| \leq \bar{\tau}, \quad \forall\ p \in \Omega \cup \partial\Omega$$

Now let $e := u - v$. Since e is a discrete function that equals 0 on $\partial\Omega$, we know from (68) that

$$\begin{aligned} \|e\|_\infty &\leq \frac{1}{8} \left\|\nabla_5^2 e\right\| \infty \\ &= \frac{1}{8} \bar{\tau} \\ &\leq \frac{h^2}{96} \left\|[\partial_x^4 + \partial_y^4] \vec{u}\right\|_\infty , \end{aligned} \tag{73}$$

Thus the method is convergent of order 2. By Lax's equivalence theorem, this means it is also stable ([5] p. 58).

2.5 Convergence of 9 point stencil

The 9 point stencil $\nabla_9^2 u_p$ is stable and of order 4, which we wish to prove in this section. We use the same strategy as with the 5 point stencil. When comparing to (62), it is easy to see that all coefficients in $-\nabla_9^2 u_p$ are positive. It is also easy to confirm that the coefficient matrix is diagonally dominant. Thus the discrete maximum principle holds for $-\nabla_9^2$ with any discrete function v defined on all points $p \in \Omega \cup \partial\Omega$ such that

$$-\nabla_9^2 v_p \leq 0.$$

In turn, this implies that any discrete function v satisfying

$$\nabla_9^2 v_p \geq 0$$

for all $p \in \Omega$ will attain its maximum on the boundary $\partial\Omega$.

This allows us to show that if v is some discrete function defined on $\Omega \cup \partial\Omega$ such that $v_p = 0$ for any $p \in \partial\Omega$, then

$$\|v\|_\infty \leq \frac{3}{40} \left\|\nabla_9^2 v\right\|_\infty .$$

Let v be a discrete function that equals 0 on the boundary. Define f as the vector of all $f_p := \nabla_9^2 v_p$. Then

$$-\|f\|_\infty \leq \nabla_9^2 v_p \leq \|f\|_\infty . \tag{74}$$

We define $w(x, y)$ as in (65). By direct calculation,

$$\begin{aligned} w(x \pm h, y + h) &= w(x, y) + \frac{1}{4}\left[2h^2 \pm (2x - 1)h + (2y - 1)h\right] \\ w(x \pm h, y - h) &= w(x, y) + \frac{1}{4}\left[2h^2 \pm (2x - 1)h - (2y - 1)h\right] \end{aligned}$$

Thus,

$$\begin{aligned} \nabla_9^2 w(x, y) &= \frac{1}{h^2}\left[\frac{1}{6}\Big(4w(x, y) + 2h^2\Big) + \frac{2}{3}\Big(4w(x, y) + h^2\Big) - \frac{10}{3}w(x, y)\right] \\ &= 1. \end{aligned}$$

By the exact same argument as in (66)-(68), we get

$$\|v\|_\infty \leq \frac{1}{8} \left\|\nabla_9^2 v\right\|_\infty . \tag{75}$$

As stated in [8, p. 328], the 9 point stencil is of order $\mathcal{O}(h^4)$. This means

$$\tau_p = Kh^4 D^6 u_p + \mathcal{O}(h^5)$$

for some constant K. Thus, defining u, v by $\nabla^2 u = f$, $\nabla_9^2 v = f$ and letting the discrete function $e := u - v$, we can use (75) together with the fact that that $e_p = 0$ on the boundary to get

$$\begin{aligned} \|e\|_\infty &\leq \frac{1}{8} \left\|\nabla_9^2 e\right\|_\infty \\ &= \frac{1}{8} \|\tau\|_\infty \\ &\leq Ch^4 \left\|D^6 u\right\|_\infty . \end{aligned}$$

Thus the method is convergent of order 4. By Lax's equivalence theorem, this means it is also stable ([5] p. 58).

2.6 Fast Poisson Solver method

The core idea of the Fast Poisson Solver (FPS) method is to exploit the symmetries of the Poisson system matrix. This section aims to shed light on how we may use this method to our advantage for the Biharmonic equation.

2.6.1 Solving $A\boldsymbol{x} = b$ with eigen decomposition

One way of solving a linear system of equations $A\boldsymbol{x} = \boldsymbol{b}$ is to find the eigenvectors $\boldsymbol{s}_i$ of A and decompose $\boldsymbol{b}$ into the eigenvector basis letting $\boldsymbol{b} = \sum_i u_i \boldsymbol{s}_i$. Then finding $\boldsymbol{x} = A^{-1}\boldsymbol{b}$ is done by letting $\boldsymbol{x} = \sum_i v_i \boldsymbol{s}_i$ and by inverting one eigenvector basis component of $\boldsymbol{b}$ at the time where $v_i = u_i/\lambda_i$, where λ_i is the eigenvalue corresponding to the eigenvector $\boldsymbol{s}_i$. To construct $\boldsymbol{x}$ we then evaluate the sum $\sum_i (u_i/\lambda_i)\boldsymbol{s}_i$.

In essence, this method is method is equivalent to finding the eigen decomposition $A = Q\Lambda Q^{-1}$ and solving $\boldsymbol{x} = A^{-1}\boldsymbol{b} = Q\Lambda^{-1}Q^{-1}\boldsymbol{b}$.

This method is generally not feasible for most matrices A, as finding eigenvalues and -vectors is an expensive computation. When A contains symmetries so that finding eigenvalues and -vectors of A (and $\boldsymbol{b}$ in the eigenvector basis of A) are cheap operations, the method might have superior performance compared to more general methods.

2.6.1.1 Eigenvectors of Kronecker products

In order to analyze the eigenvalues of the matrices we establish a few well known facts on the eigenvalues and -vectors of Kronecker product matrices.

Let A, B be matrices with eigenvectors $\boldsymbol{a}_i, \boldsymbol{b}_i$ and corresponding eigenvalues α_i, β_i. Then

$$\begin{aligned}
&(A \otimes B)(\boldsymbol{a}_i \otimes \boldsymbol{b}_j) = (A\boldsymbol{a}_i \otimes B\boldsymbol{b}_j) = (\alpha_i\boldsymbol{a}_i \otimes \beta_j\boldsymbol{b}_j) = (\alpha_i\beta_j)(\boldsymbol{a}_i \otimes \boldsymbol{b}_j). \\
\implies\ &(A \otimes A)(\boldsymbol{a}_i \otimes \boldsymbol{a}_j) = (\alpha_i\alpha_j)(\boldsymbol{a}_i \otimes \boldsymbol{a}_j) \\
\implies\ &(A \otimes I + I \otimes A)(\boldsymbol{a}_i \otimes \boldsymbol{a}_j) = (A \otimes I)(\boldsymbol{a}_i \otimes \boldsymbol{a}_j) + (I \otimes A)(\boldsymbol{a}_i \otimes \boldsymbol{a}_j) \\
&\qquad = (\alpha_i + \alpha_j)(\boldsymbol{a}_i \otimes \boldsymbol{a}_j) \qquad (76) \\
\implies\ &\left(\frac{1}{6}(A \otimes A) + A \otimes I + I \otimes A\right)(\boldsymbol{a}_i \otimes \boldsymbol{a}_j) = \left(\frac{1}{6} + 1 + 1\right)\left(\frac{1}{6}\alpha_i\alpha_j + \alpha_i + \alpha_j\right)(\boldsymbol{a}_i \otimes \boldsymbol{a}_j). \qquad (77)
\end{aligned}$$

2.6.2 Eigenvectors of the Poisson system matrices

In this chapter we find the eigenvalues and eigenvectors of our system matrices. We established in 2.3 that the eigenvectors and -values of $T' = T^{(-2)}_{N-1}$, for $k \in \{1, \dots, N-1\}$, are

$$\boldsymbol{v}_k = \left[\dots, \sin\left(\frac{\pi i k}{N}\right), \dots\right]^T, \quad i \in \{1, \dots, N-1\} \qquad (78)$$

$$\lambda_k = -2 + 2\cos(k\pi/N) \quad i \in \{1, \dots, N-1\}. \qquad (79)$$

2.6.2.1 The 5 point stencil

Using the definition of our 5 point stencil from 2.3 and $\boldsymbol{v}_k$ from above we have for a given N

$$\begin{aligned}
\nabla_5^2(\boldsymbol{v}_k \otimes \boldsymbol{v}_l) &= (T' \otimes I + I \otimes T')(\boldsymbol{v}_k \otimes \boldsymbol{v}_l) \\
&= (\lambda_k + \lambda_l)(\boldsymbol{v}_k \otimes \boldsymbol{v}_l) \\
&= \lambda^{(5)}_{k,l}(\boldsymbol{v}_k \otimes \boldsymbol{v}_l) \quad \text{where} \quad \lambda^{(5)}_{k,l} = \lambda_k + \lambda_l, \quad \lambda_i = 2 - 2\cos\left(\frac{\pi i}{N}\right) \qquad (80)
\end{aligned}$$

showing that $(\boldsymbol{v}_k \otimes \boldsymbol{v}_l),\ k, l \in \{1, \dots, N\}$ are the eigenvectors of ∇_5^2 with eigenvalues $\lambda^{(5)}_{k,l}$ ([7, p. 333]).

2.6.2.2 The 9 point stencil

Using the definition of our 9 point stencil from 2.3 and $\boldsymbol{v}_k$ from above we have for a given N

$$\begin{aligned}\nabla_9^2(\boldsymbol{v}_k \otimes \boldsymbol{v}_l) &= (\frac{1}{6}T' \otimes T' + T' \otimes I + I \otimes T')(\boldsymbol{v}_k \otimes \boldsymbol{v}_l) \\ &= (\frac{1}{6}\lambda_k\lambda_l + \lambda_k + \lambda_l)(\boldsymbol{v}_k \otimes \boldsymbol{v}_l) \\ &= \lambda_{k,l}^{(9)}(\boldsymbol{v}_k \otimes \boldsymbol{v}_l) \quad \text{where} \quad \lambda_{k,l}^{(9)} = \lambda_k\lambda_l + \lambda_k + \lambda_l, \quad \lambda_i = 2 - 2\cos\left(\frac{\pi i}{N}\right) \end{aligned} \tag{81}$$

showing that $(\boldsymbol{v}_k \otimes \boldsymbol{v}_l),\ k, l \in \{1, \ldots, N\}$ are the eigenvectors of ∇_9^2 with eigenvalues $\lambda_{k,l}^{(9)}$.

2.6.3 Eigenvectors and the discrete sine transform

The eigenvectors of the system matrices were established in the previous section as $(\boldsymbol{v}_k \otimes \boldsymbol{v}_l)_{(i,j)} = \sin(ik\pi/N)\sin(jl\pi/N),\ i, j, k, l \in \{1, \ldots, N\}$. Letting these vectors be the columns in a matrix Q and their eigenvalues along the diagonal Λ of the same size as Q and T we have $T = Q\Lambda Q^{-1}$.

The key part is as follows: Multiplying by Q^{-1} corresponds to a change of basis into the space of discrete sine functions. Multiplying by Q^{-1} is a discrete sine transform in two dimensions (the dimension of k and l). Equivalently, multiplying by Q is a 2D discrete sine transform inverse. Using this we can solve a problem $T\boldsymbol{v} = f$ by

$$T\boldsymbol{v} = Q\Lambda Q^{-1}\boldsymbol{v} = f \implies v = Q\Lambda^{-1}Q^{-1}f = \text{IDST}(\Lambda^{-1}\text{DST}(F)). \tag{82}$$

2.6.4 Problem solution

Using the methods described in this section we present the method for solving the single harmonic problem $\nabla^2 v = f$ in Algorithm 5.

2.6.5 Asymptotic runtime of solution

The algorithm descirbed will do *at least* $\mathcal{O}(N^2)$ operations per call, as the full matrix F is of size $\mathcal{O}(N^2)$ and is element-wise multiplied and added in several statements. The only operation which performs worse than $\mathcal{O}(N^2)$ is the forwards and inverse discrete sine transforms. These methods have a runtime of $\mathcal{O}(N^2 \log N^2) = \mathcal{O}(N^2 \log N)$, and are called 4 times each algorithm call.

This gives the algorithm an asymptotic runtime of $\mathcal{O}(N^2 \log N)$.

2.6.6 Alternative solution method

In this section we briefly describe an alternative solution method which has the same asymptotic runtime as the previous method, $\mathcal{O}(N^2 \log N)$, but in practice is somewhat slower in implementation. We will only consider the 9-point stencil, but the same method can be applied to the 5-point just letting $\boldsymbol{g} = \boldsymbol{f}$.

In brief, this method is based on using the discrete sine transform of each block in the T-matrix, and then solving the N separated systems of linear equations.

Mathematically we first let

$$\boldsymbol{g}_{i+(N-1)j} = \boldsymbol{g}_{(i,j)} = h^2\left[\frac{2}{3}f_{ij} + \frac{1}{12}\left(f_{i+1,j} + f_{i-1,j} + f_{i,j+1} + f_{i,j-1}\right)\right]$$

Algorithm 5 Solving the discrete biharmonic equation $\nabla^4 U = F$

procedure SolveBiharmonic(F) ▷ (Note that F is s.t. $F_{i,j} = f((i,j))$)
    $G \leftarrow \text{MakeG}(F)$
    $V \leftarrow \text{SolveHarmonic2D}(G)$

    $G \leftarrow \text{MakeG}(V)$
    $U \leftarrow \text{SolveHarmonic2D}(G)$
    return U

procedure MakeG(F)

    if 5 point stencil **then**
        return $h^2 F$

    if 9 point stencil **then**
        $G_{i,j} \leftarrow \frac{1}{12}(8F_{i,j} + F_{i+1,j} + F_{i-1,j} + F_{i,j+1} + F_{i,j-1})$ for $i, j \in \{1, \ldots, N\}$

        return $h^2 G$
procedure SolveHarmonic2D(G)
    $\lambda_k \leftarrow -2 + 2\cos(k\pi/N)$ for $k \in \{1, \ldots, N\}$

    if 5 point stencil **then**
        $E_{i,j} \leftarrow \lambda_i + \lambda_j$ for $i, j \in \{1, \ldots, N\}$

    if 9 point stencil **then**
        $E_{i,j} \leftarrow \lambda_i \lambda_j + \lambda_i + \lambda_j$ for $i, j \in \{1, \ldots, N\}$
    $\tilde{G} \leftarrow \text{DST2D}(G)$ ▷ Discrete sine transform 2D
    $\tilde{W} \leftarrow \tilde{G} \oslash E$ ▷ $\oslash$ is element-wise division
    $W \leftarrow \text{InvDST2D}(\tilde{W})$ ▷ Inverse discrete sine transform 2D
    return W

$$T\boldsymbol{u} = \frac{1}{6}\begin{bmatrix} 4T_{N-1}^{(-5)} & T_{N-1}^{(4)} & & & & \\ T_{N-1}^{(4)} & 4T_{N-1}^{(-5)} & T_{N-1}^{(4)} & & & \\ & T_{N-1}^{(4)} & 4T_{N-1}^{(-5)} & \ddots & & \\ & & \ddots & \ddots & T_{N-1}^{(4)} & \\ & & & T_{N-1}^{(4)} & 4T_{N-1}^{(-5)} & \end{bmatrix}\boldsymbol{u} = \boldsymbol{g}$$

and taking the discrete sine transform of each block matrix in T and thus each 'block' (corresponding to a row in our solution matrix) in $\boldsymbol{u}$ and $\boldsymbol{g}$ we have

$$\begin{bmatrix} \Lambda & \Lambda' & & & \\ \Lambda' & \Lambda & \Lambda' & & \\ & \Lambda' & \Lambda & \ddots & \\ & & \ddots & \ddots & \Lambda' \\ & & & \Lambda' & \Lambda \end{bmatrix}\begin{bmatrix} \tilde{\boldsymbol{u}}_1 \\ \tilde{\boldsymbol{u}}_2 \\ \tilde{\boldsymbol{u}}_3 \\ \vdots \\ \tilde{\boldsymbol{u}}_{N-1} \end{bmatrix} = \begin{bmatrix} \tilde{\boldsymbol{g}}_1 \\ \tilde{\boldsymbol{g}}_2 \\ \tilde{\boldsymbol{g}}_3 \\ \vdots \\ \tilde{\boldsymbol{g}}_{N-1} \end{bmatrix}.$$

We now have a series of equations

$$\Lambda'\tilde{\boldsymbol{u}}_{i-1} + \Lambda\tilde{\boldsymbol{u}}_i + \Lambda'\tilde{\boldsymbol{u}}_{i+1} = \tilde{\boldsymbol{g}}_i$$

(with edge-cases $\tilde{\boldsymbol{u}}_0 = \tilde{\boldsymbol{u}}_{N-1} = \boldsymbol{0}$). Observe that elements j and k, $j \neq k$, in each of the row-vectors $\tilde{\boldsymbol{u}}_i$, are separated and do not depend on each other, only on the elements in the same position in the other vectors. Using this fact we separate out each of the column-vectors in $\tilde{\boldsymbol{u}}$ and $\tilde{\boldsymbol{v}}$ (viewed as matrices), naming them $\tilde{\boldsymbol{p}}$ and $\tilde{\boldsymbol{q}}$ and get the separated equations

$$L_j\tilde{\boldsymbol{p}}_j = \tilde{\boldsymbol{q}}_j \quad \text{for } j \in \{1, \dots N-1\}$$

where L_j is a TST-matrix with $\lambda_j = \frac{4}{6}(-5 + 2\cos(j\pi/N))$ on the diagonal and $\lambda_j = \frac{1}{6}(4 + 2\cos(j\pi/N))$ on the off-diagonal (for the 9-point stencil). These $N-1$ equations can be solved efficiently with the discrete sine transform method corresponding to the aforementioned eigen decomposition. The $N-1$ calls to the solver cost $\mathcal{O}(N^2 \log N)$ operations in total.

When these solutions $\tilde{\boldsymbol{p}}$ are found we combine them in the same way we separated $\tilde{\boldsymbol{u}}$ and then transform them back using the row-wise inverse discrete transform to find our solution $\boldsymbol{u}$.

This method can be implemented using Algorithm 6 defined below and simply exchanging 'Solve-Harmonic2D' with 'SolveHarmonic' in 5. This mehtod is also implemented in our code and named the 'rowwise Fourier', contrasting the '2D Fourier' described in the previous section.

It might be worth noting that this process can be paralellized, giving a runtime of $\mathcal{O}(N^2 + (N^2 \log N)/p)$ when using p parallel processes.

2.7 UMR on 5 and 9 point stencils

In order to verify the order of the 5- and 9-point stencils, we construct a manufactured solution to problem (46) and perform uniform mesh refinement on it. As indicated in section 2.1, if we choose

$$f(x, y) = \sin(\pi x)\sin(\pi y), \tag{83}$$

the function

$$u(x, y) = \frac{f(x, y)}{4\pi^4} \tag{84}$$

solves problem (46). These functions are plotted in Figure 20. From the order analysis in sections 2.4 and 2.5, we expect order $\mathcal{O}(h^2)$ and $\mathcal{O}(h^4)$ respectively for the 5- and 9- point stencils. This

Algorithm 6 Solving the discrete harmonic equation $\nabla^2 U = F$ using an alternate method

procedure SolveHarmonic(G)

  if 5 point stencil **then**
    $\lambda_j \leftarrow -4 + 2\cos(j\pi/N)$ for $j \in \{1, \dots, N\}$
    $\lambda'_j \leftarrow 1$ for $j \in \{1, \dots, N\}$

  if 9 point stencil **then**
    $\lambda_j \leftarrow \frac{4}{6}(-5 + 2\cos(j\pi/N))$ for $j \in \{1, \dots, N\}$
    $\lambda'_j \leftarrow \frac{1}{6}(4 + 2\cos(j\pi/N))$ for $j \in \{1, \dots, N\}$
  $\tilde{G} \leftarrow$ row-wise DST(G)
  for $j \in \{1, \dots, N-1\}$ **do**
    $\tilde{\boldsymbol{q}} \leftarrow$ row j in $\tilde{G}$
    $\tilde{\boldsymbol{p}} \leftarrow$ Solve_TST_With_DST($\tilde{\boldsymbol{q}}, \lambda_j, \lambda'_j$) ▷ The λs are the on and off-diagonals in the TST
    Column j in $\tilde{U} \leftarrow \tilde{p}$
  $U \leftarrow$ row-wise InvDST($\tilde{G}$)
  return U

translates to $\mathcal{O}(\text{Ndof})$ and $\mathcal{O}(\text{Ndof}^2)$, respectively. Figure 21 confirms this, plotting the relative error for $M = \{8, 16, 32, 64, 128, 256\}$ using both stencils.

2.8 Solving the Biharmonic equation efficiently

We now consider the Biharmonic equation (46) with the manufactured solution

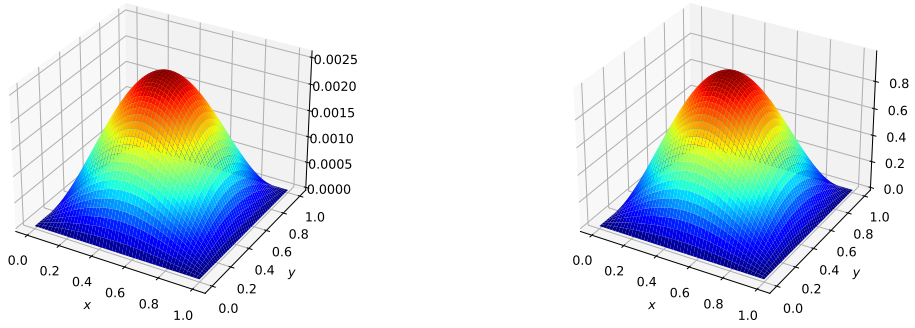
$$u(x, y) = (\sin(\pi x)\sin(\pi y))^4 \exp(-(x - 0.5)^2 - (y - 0.5)^2). \tag{85}$$

The function f will simply be defined as

$$f(x, y) := \nabla^4 u(x, y), \tag{86}$$

since the actual expression is quite cumbersome to write out. Both (85) and (86) are plotted in figure 22. Using the 9 point stencil together with the Fast Poisson Solver method described in section 2.6, we compute both the convergence and time consumption of our solver. The results for $M = 2^s$, $s = 3, \dots, 13$ are plotted in Figure 23.

We observe the $\mathcal{O}(h^2)$ and $\mathcal{O}(h^4)$ error convergence of the 5 point and 9 point stencils respectively, as well as the tendency towards the theoretical asymptotic runtime $\mathcal{O}(N^2 \log N) = \mathcal{O}(Ndof \log Ndof)$. Our preferred method, the '2D fourier', is in general faster than the row wise Fourier. There is some difference between the 5 and 9 point stencils in execution time.



(a) $u(x, y)$ as defined in (84).

(b) $f(x, y)$ as defined in (83).

Figure 20: Functions solving the biharmonic equation (46).

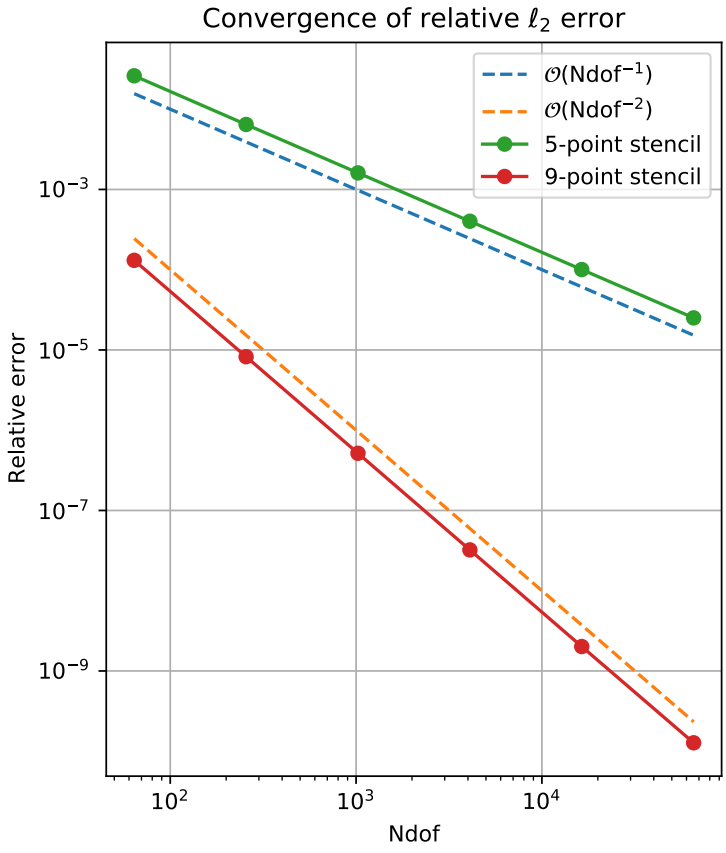
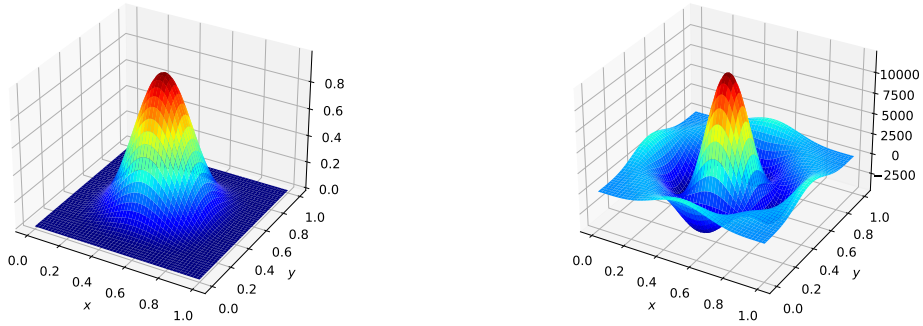


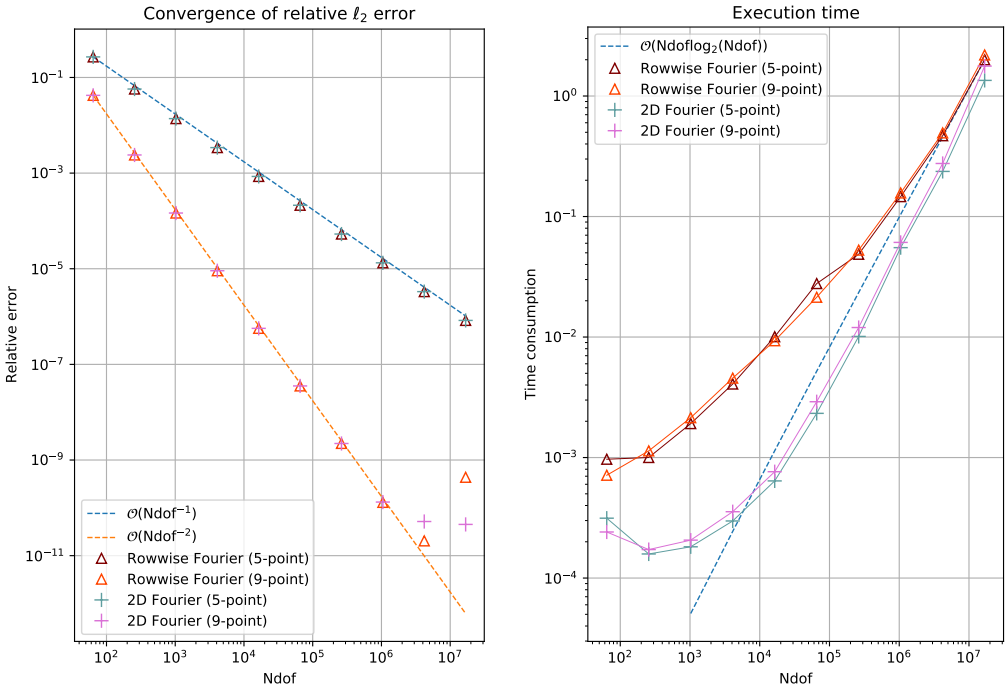
Figure 21: Numerical solutions of problem (46) using different stencils, with f as defined in (83). Relative error measured using the ℓ_2 norm.



(a) $u(x, y)$ as defined in (85).

(b) $f(x, y)$ as defined in (86).

Figure 22: Functions solving the Biharmonic equation (46).



(a) Relative error

(b) Time consumption

Figure 23: Convergence and runtime of different methods for solving the Biharmonic equation (46) with f as defined in (86). Relative error measured in the ℓ_2 norm, execution time measured in seconds. The different methods are outlined in 2.6 and described both in analysis and pseudocode.

Bibliography

[1] Fausto Saleri Alfio Quarteroni Riccardo Sacco. *Numerical Mathematics*. Vol. Second edition. Springer, 2006. ISBN: 978-3-540-34658-6.

[2] Mingkui Chen. *On The Solution of Circulant Linear Systems*. 1985.

[3] Charles Curry. *TMA4212 Part 2: Introduction to finite element methods*. 2018.

[4] Kamy Sepehrnoori Liu Jianchun Gary A. Pope. 'A high-resolution finite-difference scheme for nonuniform grids'. In: Volume 19, Issue 3 (1995), pp. 162–172. ISSN: 0307-904X. DOI: 10.1016/0307-904X(94)00020-7.

[5] B. Owren. *TMA4212 numerical solution of partial differential equations with finite difference methods*. 2017.

[6] Lothar Reichel Silvia Noschese Lionello Pasquini. *Tridiagonal Toeplitz Matrices: Properties and Novel Applications*. 2006.

[7] Gilbert Strang. *Mathematical Methods for Engineers II*. 2005.

[8] John C. Strikwerda. *Finite Difference Schemes and Partial Differential Equations*. Vol. Second edition. Siam, 2004. ISBN: 0-89871-567-9.