

1 Poisson equation in one dimension

In this section, we will solve the one-dimensional Poisson equation

$$u_{xx} = f(x) \quad (0 < x < 1) \quad (1)$$

subject to a source term $f(x)$ and different boundary conditions at $x = 0$ and $x = 1$. First, we will solve it with finite difference methods of first and second order on a uniform grid. Finally, we solve it on a non-uniform grid and investigate how adaptive mesh refinement (AMR) can be used to obtain accurate solutions by distributing fewer points more cleverly along the grid.

1.1 Analytical solution

One way to express the analytical solution is to simply integrate equation (1) twice to get

$$\begin{aligned} u(x) &= C_1 + \int^x dx' u_x(x') \\ &= C_1 + \int^x dx' \left(C_2 + \int^{x'} dx'' u_{xx}(x'') \right) \\ &= C_1 + C_2 x + \int^x dx' \int^{x'} dx'' f(x''), \end{aligned} \quad (2)$$

where the constants C_1 and C_2 are determined from two boundary conditions and the integrals can be done from any lower limit. Note that this is equivalent to saying that the solution is a sum of the solution to the homogenous equation $u_{xx} = 0$ and a solution to the inhomogenous equation $u_{xx} = f(x)$.

Note that if *two* Neumann boundary conditions $u_x(0) = a$ and $u_x(1) = b$ are imposed, then the solution $u(x)$ is unique only up to a constant. If $u(x)$ is a solution to the boundary value problem defined by equation (1) and $u_x(0) = a$ subject to $u_x(1) = b$, then also $(u + C)_{xx} = u_{xx}$ in the interior and $(u + C)_x(0) = u_x(0) = a$ on the left boundary, and similarly on the right boundary $x = 1$. It can also be seen by observing that C_1 is undetermined when 2 is differentiated.

1.2 Numerical solution on a uniform grid

First, consider the boundary value problem defined by equation (1), subject to the boundary conditions

$$u(0) = a \quad \text{or} \quad u_x(0) = a \quad \text{and} \quad u(1) = b \quad \text{or} \quad u_x(1) = b.$$

To solve the equation numerically, we divide the interval $[0, 1]$ into the uniform grid

$$\begin{array}{ccccccc} x_0 = 0 & x_1 & x_2 & \dots & x_m & \dots & x_{M-1} & x_M & x_{M+1} = 1 \\ \bullet & \bullet & \bullet & \dots & \bullet & \dots & \bullet & \bullet & \bullet \\ & h & h & & & & h & h & \end{array}$$

of $M + 2$ points and step length h . We approximate the second derivative at interior points with the central difference

$$\frac{\partial^2 u}{\partial x^2}(x_m) = \frac{u_{m-1} - 2u_m + u_{m+1}}{h^2} + \mathcal{O}(h^2) \quad (1 \leq m \leq M - 1).$$

To handle the Dirichlet boundary condition $u(0) = a$ at the left edge or $u(1) = b$ at the right edge, we insert the trivial equation

$$1 \cdot u_0 = a \quad \text{or} \quad 1 \cdot u_{M+1} = b.$$

To handle the Neumann boundary condition $u_x(0) = a$ at the left edge or $u_x(1) = b$ at the right edge to second order, we approximate the first derivative to second order with forward or backward differences to

get

$$u_x(0) = \frac{-\frac{3}{2}u_0 - 2u_1 - \frac{1}{2}u_2}{h} + \mathcal{O}(h^2) = b \quad \text{or} \quad u_x(1) = \frac{\frac{1}{2}u_{M-1} - 2u_M + \frac{3}{2}u_{M+1}}{h} + \mathcal{O}(h^2) = b.$$

Writing all these equations in $(M+2) \times (M+2)$ -matrix form $AU = b$, we obtain for example with $u(0) = a$ and $u_x(1) = b$

$$\begin{bmatrix} 1 & & & & \\ +1/h^2 & -2/h^2 & +1/h^2 & & \\ & \ddots & \ddots & \ddots & \\ & & +1/h^2 & -2/h^2 & +1/h^2 \\ & & +1/2h & -2/h & +3/2h \end{bmatrix} \begin{bmatrix} U_0 \\ U_1 \\ \vdots \\ U_M \\ U_{M+1} \end{bmatrix} = \begin{bmatrix} a \\ f(x_1) \\ \vdots \\ f(x_M) \\ b \end{bmatrix}, \quad (3)$$

where the first and last rows of the matrix generally vary depending on the boundary conditions.

Note that if the numerical solution is subject to two Neumann boundary conditions, the matrix becomes singular and the solution non-unique. In this case, we impose the additional constraint $U_0 = 0$ by setting all entries in the first column of A to zero. To handle the singular matrix, we solve the system with a least-squares method instead of a GESV LU factorization method.

We now apply our method to the boundary value problem with the source function

$$f(x) = x + \cos(2\pi x).$$

Inserting it into equation (2) and doing the integrals, we get the exact solution

$$u(x) = C_1 + C_2x + \frac{1}{3!}x^3 - \frac{1}{4\pi^2} \cos(2\pi x).$$

In figure 1, we present numerical solutions for three different combinations of boundary conditions.

Our approach to handling the boundary conditions is not the only possible approach. The system of equations is equivalent if we remove the first row and column of A and the first entries in U and b , but simultaneously modify the entry $f(x_1) \rightarrow f(x_1) - a/h^2$. This approach is more consistent with treating U_0 as a known variable, since its precise value is defined by the Dirichlet boundary condition. However, our approach of inserting a trivial equation $1 \cdot U_0 = a$ keeps the matrix dimensions independent of boundary conditions and makes it easier to reason with how the discretized differential operator represented by A operates on the grid point U_0 in the same way it operates on all other grid points.

Neumann boundary conditions can also be handled differently. Instead of approximating the second derivative only on actual grid points, we could approximate it with a fictitious point x_{-1} and a central difference $u_x(0) \approx (U_1 - U_{-1})/(2h)$. Then we could use this together with the central difference $(U_{-1} - 2U_0 + U_1)/h^2 = f(x_0)$ to eliminate U_{-1} . Eliminating U_{-1} , the first equation becomes $(U_1 - U_0)/h = a + hf(x_0)/2$, so the boundary condition could be handled by setting the first row to $[-1/h, +1/h, 0, \dots]$ and modifying the first entry in b to $a \rightarrow a + hf(x_0)/2$. This would also be second order, and would allow us to use the same stencil also at x_0 , but we would then have to pay the price of modifying the right side of the matrix equation in an unnatural way.

1.3 Adaptive numerical solution on a non-uniform grid

We will now demonstrate how the numerical solution can be generalized to a non-uniform grid with $x_i - x_{i-1} \neq \text{const}$. Then we will attempt to make the numerical solution as good as possible using as few grid points as possible, by placing points tighter where the solution varies rapidly.



Figure 1: The left plots show analytical solutions $u(x)$ and numerical solutions $U(x)$ with $M = 30$ grid points for $u_{xx} = x + \cos 2\pi x$ subject to three different boundary conditions. The right plots show convergence plots corresponding to the same boundary conditions, where the error is measured with both the a continuous and discrete L_2 -norm. TODO: add triangle with slope in convergence plot

$$\begin{aligned}
u_m'' &\approx \frac{u'_{m+1/2} - u'_{m-1/2}}{x_{m+1/2} - x_{m-1/2}} \\
&= \frac{2}{x_{m+1} - x_{m-1}} \left(u'_{m+1/2} - u'_{m-1/2} \right) \\
&= \frac{2}{x_{m+1} - x_{m-1}} \left(\frac{u_{m+1} - u_m}{x_{m+1} - x_m} - \frac{u_m - u_{m-1}}{x_m - x_{m-1}} \right)
\end{aligned} \tag{4}$$

Assuming Dirichlet boundary conditions, we then write the nonzero entries of the matrix A (indexed from zero) as

$$\begin{aligned}
A_{00} &= A_{MM} = 1 \\
A_{m,m-1} &= \frac{2}{x_{m+1} - x_{m-1}} \frac{1}{x_m - x_{m-1}} \\
A_{m,m} &= \frac{-2}{x_{m+1} - x_{m-1}} \left(\frac{1}{x_m - x_{m-1}} + \frac{1}{x_{m+1} - x_m} \right) \\
A_{m,m+1} &= \frac{2}{x_{m+1} - x_{m-1}} \frac{1}{x_{m+1} - x_m}.
\end{aligned}$$

The job is then again to solve the system $AU = b$. Note that the stencil reduces to the one in equation (3) when $x_m - x_{m-1} = x_{m+1} - x_m = h$.

To do adaptive mesh refinement, we will

1. Start with a coarse grid with uniform spacing, such as $[x_0, x_1] = [0, 1]$.
2. Wisely choose *one* grid interval $[x_m, x_{m+1}]$ based on some strategy.
3. Split the interval in half by inserting a new point at $(x_m + x_{m+1})/2$.
4. Repeat step 2 and 3 until the grid has the desired resolution.

We will compare three different strategies for selecting the grid interval:

1. **Error strategy:** Select the interval $[x_m, x_{m+1}]$ with the largest error

$$\int_{x_m}^{x_{m+1}} dx |u(x) - U(x)|, \quad \text{where } U(x) = U_m + \frac{x - x_m}{x_{m+1} - x_m} (U_{m+1} - U_m)$$

is a linearly interpolated numerical solution on the *current* grid and $u(x)$ is the exact solution. This strategy requires knowledge of the exact solution $u(x)$ and solving the system numerically before each splitting.

2. **Truncation error strategy:** Select the interval $[x_m, x_{m+1}]$ with the largest absolute truncation error

$$\left| \frac{2}{x_{m+1} - x_m} \left(\frac{u_{m+1} - u_{m+1/2}}{x_{m+1} - x_{m+1/2}} - \frac{u_{m+1/2} - u_m}{x_{m+1/2} - x_m} \right) - f(x_m) \right|,$$

upon insertion of a middle point $x_{m+1/2} = (x_m + x_{m+1})/2$, where $u(x)$ is the exact solution. This strategy also requires knowledge of the exact solution $u(x)$, but does not rely on intermediate computations of the numerical solution U_m .

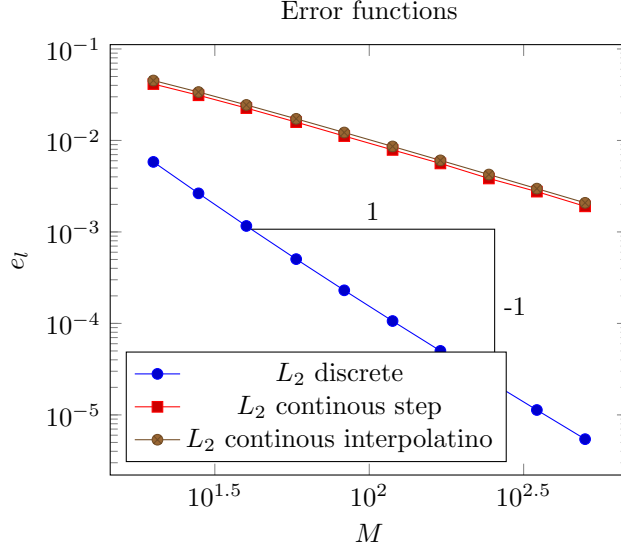


Figure 2: During adaptive mesh refinement (AMR) with the error strategy (strategy number 1), the interval with the largest error $\int dx |u(x) - U(x)|$ is split in half. Here, $u(x) = \exp(-(x - 1/2)^2/0.1)$ is the symmetric solution to whichever Poisson equation has $f(x) = u_{xx}$ on $x \in [0, 1]$. Symmetry is imposed numerically by also adding the point $1 - x$ to the grid whenever a point $x \neq 1/2$ is added.

3. **Source strategy:** Select the interval $[x_m, x_{m+1}]$ with the largest “absolute source”

$$\int_{x_m}^{x_{m+1}} dx |f(x)|.$$

In physical applications, $f(x)$ is typically mass density or charge density. The idea is to refine intervals on which there is much mass or charge, as the solution is expected to vary faster there. This splitting strategy requires neither knowledge of the exact solution or the numerical solution, only on the source function $f(x)$, as is typically the case in practice.

In figure 2, we demonstrate how the initial grid $[0, 0.5, 1]$ and the numerical solution $U(x)$ evolves through adaptive refinement with the error strategy. Observe how the refinement concentrates on resolving critical areas of the solution near the peak and the inflection points.

In figure 3, we compare the convergence of the three adaptive refinement strategies to the $\mathcal{O}(h^2)$ -convergence of uniform refinement on the same problem. The error strategy requires knowledge of the exact solution and intermediate computations, but in return it is the most effective strategy. The source strategy requires neither, but is also the least effective strategy. We can say that the more knowledge of the exact solution and intermediate computations, the greater the accuracy.

Note that the errors are not strictly decreasing with each refinement $M \rightarrow M + 2$. In particular, the error from the error strategy exhibits an oscillating pattern for $M \geq 32$. This is a weakness of refining only *exactly* two symmetric intervals for each refinement. An alternative method is to refine *multiple* intervals at every refinement step using a criterion that splits not only the interval with the largest error, but all intervals with error above some reference error. This procedure removes our control over the exact number of intervals, but in return gives us control over the maximal acceptable error on any interval. In section 6.4.2, we will improve our AMR strategy exactly in this way. The effect is that the oscillating pattern is eliminated and that the error decreases strictly with each refinement step. This will be equivalent to jumping directly from one local minimum in the oscillation to the next.

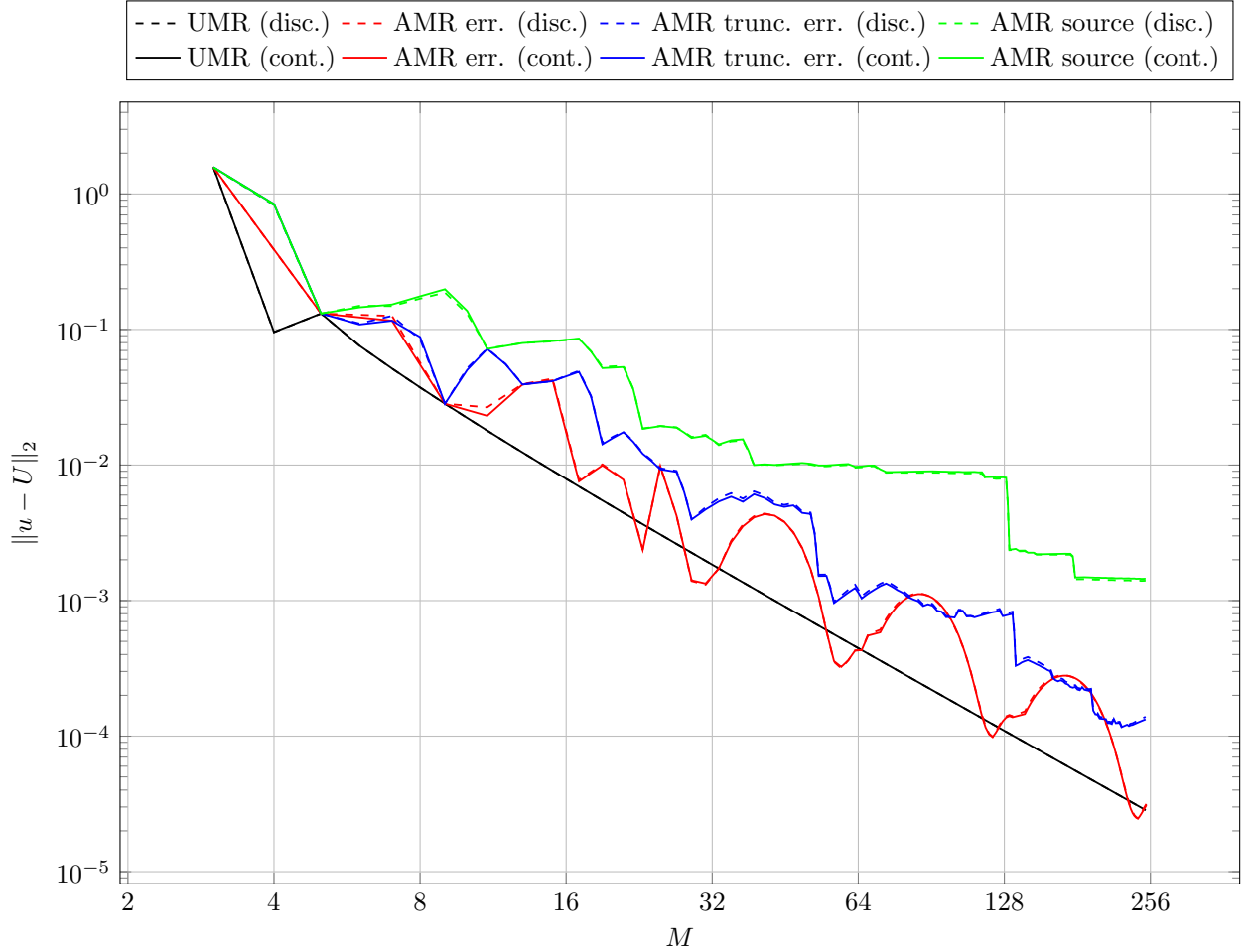


Figure 3: Comparison between the convergence of the numerical solution $U(x)$ with uniform mesh refinement (UMR) and adaptive mesh refinement (AMR) on the problem $u_{xx} = f(x)$ on $x \in [0, 1]$ with analytical solution $u(x) = \exp(-(x - 1/2)^2/0.1)$. The adaptive refinement is done using three different strategies that subdivide the interval with the largest absolute error $|u - U|$, largest truncation error $Lu - f(x)$ (where $L \approx \partial^2/\partial x^2$ is the discretized differentiation operator) or largest amount of source $\int dx |f(x)|$. Errors $\|u - U\|_2$ are measured with the continuous and discrete L_2 -norm.

2 Heat equation in one dimension

In this section, we consider the one-dimensional heat equation for $u = u(x, t)$,

$$u_t = u_{xx}, \quad u(x, 0) = f(x), \quad x \in [0, 1] := \Omega,$$

with either Neumann or Dirichlet boundary conditions, and solve it numerically using both the Backward Euler method and Crank-Nicolson method. These are $\mathcal{O}(k + h^2)$ and $\mathcal{O}(k^2 + h^2)$ methods respectively, and we will analyze and compare their convergence using mesh refinement as we did in section 1. However we will here restrict our attention to uniform grids only.

2.1 Numerical solution method

To solve the heat equation numerically we first perform semi-discretization, i.e. we do spatial discretization and keep the time continuous. The interval Ω is divided into M equidistant points with separation $h = 1/(M - 1)$, as described in section 1, and we express the spatial derivative using the central finite difference to get

$$u_t(x_m, t) = \frac{1}{h^2} \delta_x^2 u(x_m, t) + \mathcal{O}(h^2), \quad m = 1, \dots, M.$$

We now introduce the single variate functions $v_m(t)$ as the approximation to $u(x_m, t)$, at each x_m , turning the PDE into a set of ODEs

$$\frac{dv_m(t)}{dt} = \frac{1}{h^2} \delta_x^2 v_m(t), \quad v_m(0) = f(x_m).$$

The problem is then generally solved by imposing the boundary conditions, and numerically integrating the equations in time. The integration can be done by using one of the many off-shelf routines for ODEs, e.g. Euler's method, and for convenience we employ the θ -method, which for general ODEs $y' = g(y, t)$ is given as

$$y^{n+1} = y^n + \Delta t((1 - \theta)g(y^n, t_n) + \theta g(y^{n+1}, t_{n+1})).$$

Using a constant time step $\Delta t = k$ leads to uniform discretization of the time interval, which we divide into N equidistant points, and the final discrete grid for space and time is illustrated in figure 4. This gives the approximate solution of $v_m(t)$ at N finite times $t_n = nk$, $n = 1, \dots, N$, $k = 1/(N - 1)$, and we finally denote the fully discretized approximation of $u(x_m, t_n)$ as U_m^n . The θ -method for the 1D heat equation is then written out as

$$\text{thetamethodforheatequation}, \tag{5}$$

and setting the value of θ in (5) determines the specific numerical scheme. We have

$$\begin{aligned} \text{Forward Euler} \quad \theta &= 0 \\ \text{Backward Euler} \quad \theta &= 1 \\ \text{Crank-Nicolson} \quad \theta &= \frac{1}{2}, \end{aligned}$$

and we will as mentioned consider the Backward-Euler and Crank-Nicolson methods.

Imposing boundary conditions and some matrix form/equations here I guess.

With the numerical schemes in hand we now solve the heat equation with the following Neumann boundary conditions and initial condition,

$$u_x(0, t) = u_x(1, t) = 0, \quad u(x, 0) = 2\pi x - \sin(2\pi x). \tag{6}$$

The computed solutions for $t \in [0, 0.5]$ is plotted in figure 5, and qualitatively the solution behaves in accordance to what one should expect for the heat equation. To quantify and compare the accuracy of the numerical schemes we will now proceed to analyze convergence using mesh refinement, similar to what we did in section 1.

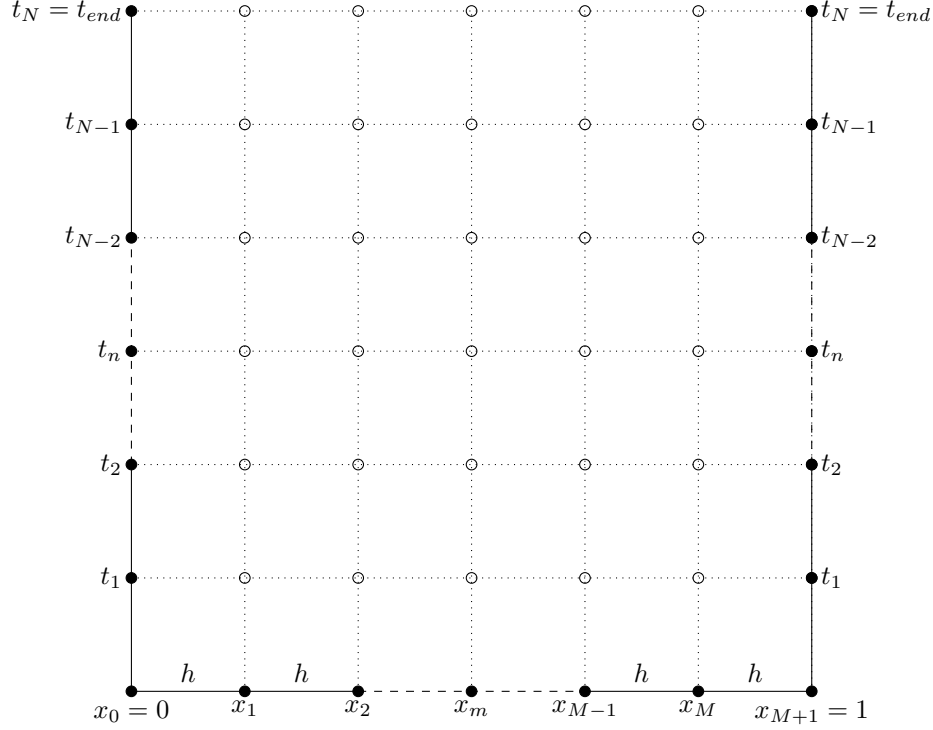


Figure 4: Discrete uniform grid

2.2 Convergence and mesh refinement

As in section 1 we now analyze the convergence of the numerical solution methods, by doing mesh refinement of the spatial grid x_m . For (6), the analytical solution is not available in closed form, so in order to analyze convergence we compute a reference solution using a sufficiently high M , which we use in place of the analytical solution when computing the error.

In order to analyze the convergence further, we also consider the heat equation with a set of boundary and initial conditions for which the analytical solution is known. Specifically we consider

$$u(0, t) = u(1, t) = 0, \quad u(x, 0) = \sin(\pi x), \quad (7)$$

on the same domain $x \in [0, 1] := \Omega$ and $t > 0$. Note that we now have Dirichlet boundary conditions, and the analytical solution is readily available as ¹

$$u(x, t) = \sin(\pi x)e^{-\pi^2 t}. \quad (8)$$

When doing mesh refinement of the spatial grid, we vary the number of spatial grid points M , and compute the numerical solution at the same point in time $t = t_{end}$. We keep the number of time steps N is kept fixed, so that the error from the time discretization does not vary, and compute the L_2 discrete relative error with respect to the reference solution. For equation (7) we also compute the l_2 continuous relative error, and compute errors with respect to the analytical solution. The resulting convergence plots are shown in figure ?? and ??.

Both methods are second order in the spatial step h , but Crank-Nicolson is one order higher in time step k .

¹The analytical solution can be computed using e.g. separation of variables.

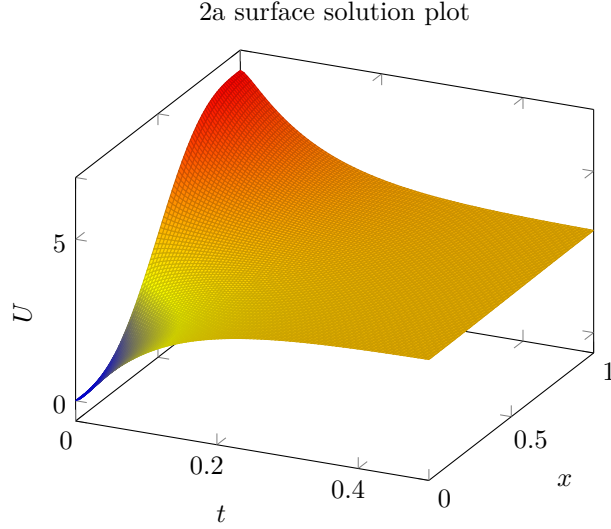


Figure 5: I am a surface plot, Hooray :)

3 Inviscid Burgers' equation

In this section we turn to solve the inviscid Burgers' equation with given Dirichlet boundary conditions and initial condition

$$u_t = -uu_x, \quad u(0, t) = u(1, t) = 0, \quad u(x, 0) = \exp(-400(x - 1/2)^2). \quad (9)$$

This equation exhibits breaking; after some point in time t_b the solution breaks, and the unique solution does not exist, leading to the formation of a *shock wave*.^[?] The time t_b before this can happen is given by

$$t_b = \frac{-1}{\min f'(x)}, \quad (10)$$

where $f(x)$ is the given initial condition $u(x, 0) = f(x)$.^[?]

Numerical solution method

To solve (9) numerically we perform semidiscretization in the same way as we did for the heat equation in section 2, also on a uniform spatial grid as described in section 1. The resulting system of ODEs is

$$\text{burgersdiscretized},$$

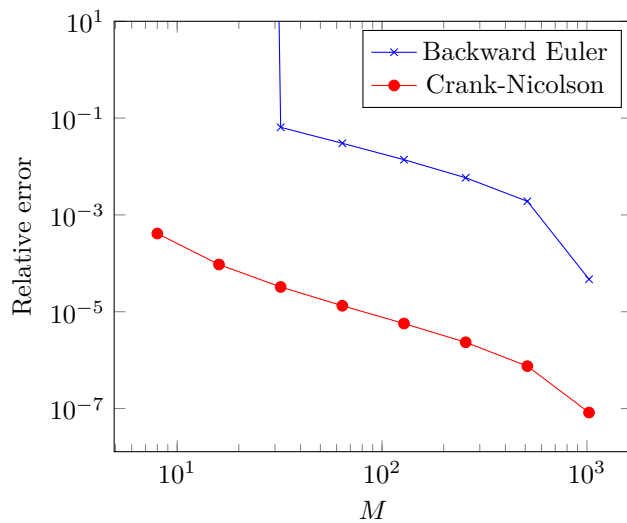
which we integrate using `solve_ivp` from the SciPy library, with the default explicit Runge-Kutta method of order 4(5)^[?].

Comment/question: Is it fine to use `scipy.integrate.solve_ivp`, or should it be all home cooking?

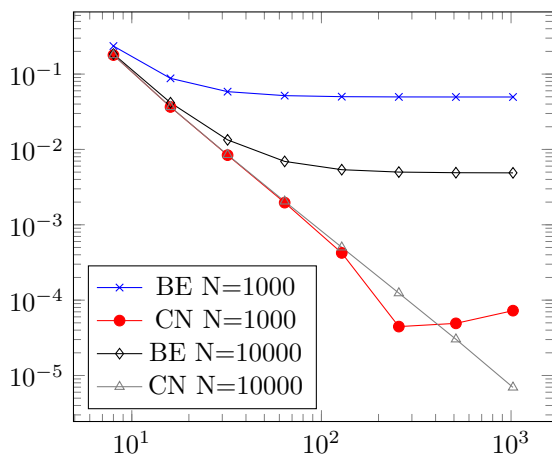
3.1 Time of breaking

Insertion of $u(x, 0)$ for f in (10), gives $t_b \approx 0.058$. To get a criterion for when the numerical solution has broken down, we use that the stable solution should be strictly increasing from $x = 0$ to towards the apex, and then strictly decreasing from the apex towards the right boundary at $x = 1$. When this is no longer the case we say that the solution has broken, and the time for which this happened for our solution was at $t_* \approx 0.055$. Figure ?? shows the solution sampled around the time of breaking.

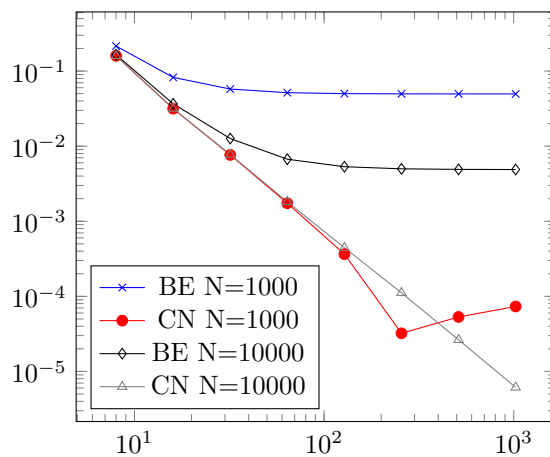
2a Convergence plot L2 discrete rel. error

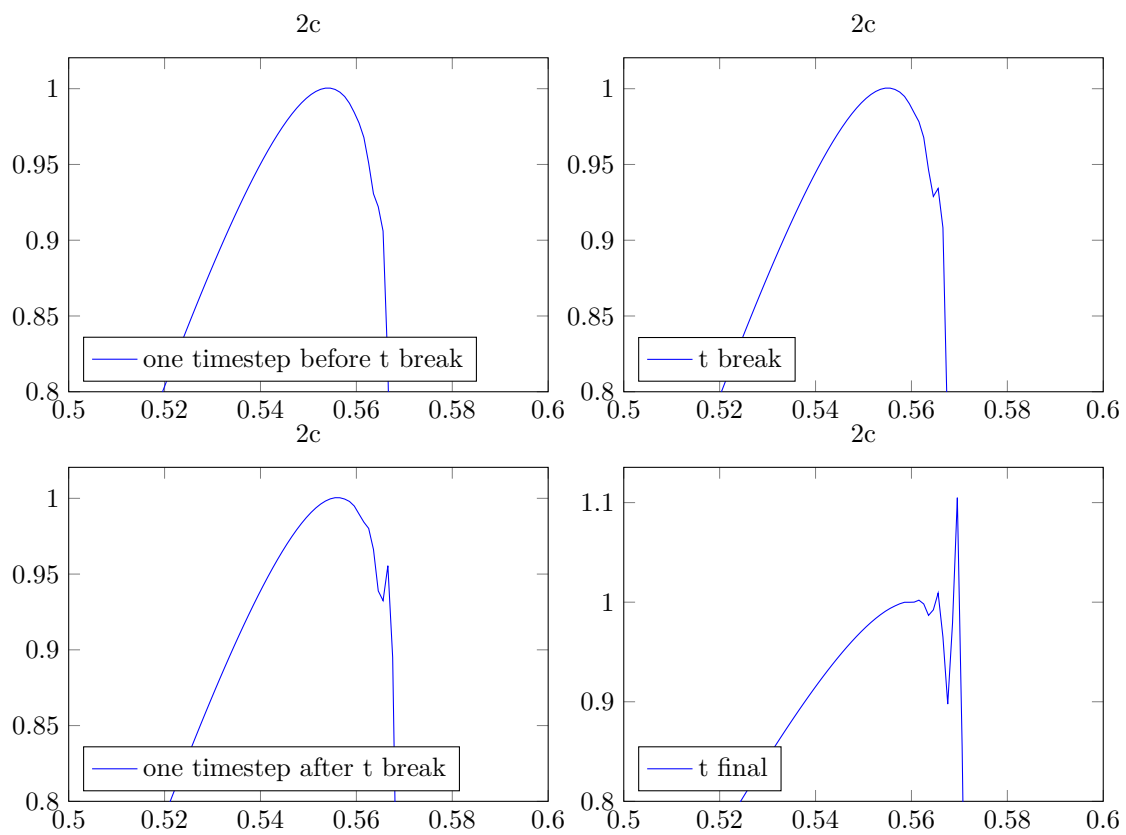


2b discrete



2b continuous





4 Laplace equation in two dimensions

In this section, we will solve the two-dimensional Laplace equation on a quadratic domain

$$u_{xx} + u_{yy} = 0, (x, y) \in \Omega := [0, 1]^2, \quad (11)$$

with boundary conditions on the edges of Ω

$$\begin{aligned} u(0, y) &= 0, \\ u(1, y) &= 0, \\ u(x, 0) &= 0, \\ u(x, 1) &= \sin(2\pi x). \end{aligned}$$

We will solve this equation numerically using a five point stencil, but first, we solve it analytically to provide a reference solution which can be compared with the numerical one.

4.1 Analytical solution

The solution of equation 11 can be found by separation of variables. First, assume that we can write

$$u(x, y) = \alpha(x)\beta(y),$$

which implies that

$$u_{xx} + u_{yy} = \alpha''(x)\beta(y) + \alpha(x)\beta''(y) = 0,$$

where the prime markers ' denote differentiation of the single variable functions $\alpha(x)$ and $\beta(y)$. Rearranging, we get that

$$\frac{\alpha''(x)}{\alpha(x)} = \frac{\beta''(y)}{\beta(y)} = c$$

must be constant, since α and β are functions of independent variables. Thus, we have two second order differential equations

$$\begin{aligned} \alpha''(x) - c\alpha(x) &= 0, \\ \beta''(y) - c\beta(y) &= 0, \end{aligned}$$

with boundary conditions

$$\begin{aligned} \alpha(0) = \alpha(1) = \beta(0) &= 0, \\ \alpha(x)\beta(1) &= \sin(2\pi x). \end{aligned}$$

Setting $\beta(1)$ to 1 yields $\alpha(x) = \sin(2\pi x)$, so that $\alpha''(x) = -4\pi^2\alpha(x)$ where $y = 1$, we find that $c = -4\pi^2$. Solving the equation for $\beta(y)$, we find that

$$\beta(y) = b_1 e^{\sqrt{c}y} + b_2 e^{-\sqrt{c}y}.$$

Inserting $c = -4\pi^2$ and the boundary conditions $\beta(0) = 0$ and $\beta(1) = 1$, we get

$$\beta(y) = \frac{\sinh(2\pi y)}{\sinh(2\pi)},$$

and finally

$$u(x, y) = \frac{\sin(2\pi x) \cdot \sinh(2\pi y)}{\sinh(2\pi)}.$$

4.2 Numerical solution

Numerically, we can solve this equation by dividing the domain Ω into a grid. That is, we divide the interval $x \in [0, 1]$ and $y \in [0, 1]$ into M and N parts, respectively, so that the domain contains $M * N$ points, in which we will approximate the solution to equation 11.

Rewriting Laplace's equation using central differences, we get

$$\begin{aligned}\partial_x^2 u(x_m, y_n) &= \frac{1}{h^2} [u(x_{m-1}, y_n) + 2u(x_m, y_n) + u(x_{m+1}, y_n)] + \mathcal{O}(h^2) \\ &= \frac{1}{h^2} \delta_x^2 u(x_m, y_n), \\ \partial_y^2 u(x_m, y_n) &= \frac{1}{k^2} [u(x_m, y_{n-1}) + 2u(x_m, y_n) + u(x_m, y_{n+1})] + \mathcal{O}(k^2) \\ &= \frac{1}{k^2} \delta_y^2 u(x_m, y_n),\end{aligned}$$

where (x_m, y_n) denote the point (m, n) in the grid. Adding these expressions, and naming our approximated solution $U_{m,n} := u(x_m, y_n)$, we find that the Laplace equation can be approximated

$$0 = \partial_x^2 u(x_m, y_n) + \partial_y^2 u(x_m, y_n) \approx \frac{1}{h^2} \delta_x^2 U_{m,n} + \frac{1}{k^2} \delta_y^2 U_{m,n},$$

or, simplifying the notation with the notation visualized in figure ??,

$$\frac{1}{k^2} (U_{above} + U_{below} - 2U_{center}) + \frac{1}{h^2} (U_{left} + U_{right} - 2U_{center}) = 0.$$

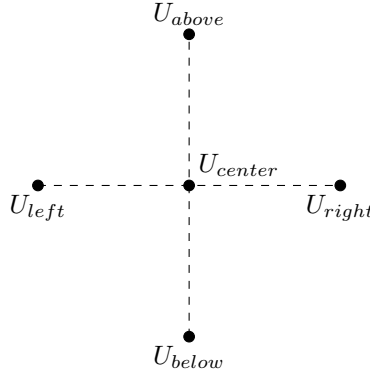


Figure 6: The five-point stencil corresponding to central difference differentiation in both the x - and y -direction.

This stencil can be used to approximate the value of $U(x_m, y_n) = U_{center}$ for all points (x_m, y_n) in the grid. Ignoring firstly the above and below nodes of the stencil, we can easily set up a matrix A' in the same way as in Section 1. Note that this is done only in order to clarify the derivation – the matrix A' is merely a "stepping stone" – not a useful result.

$$A'U = \frac{1}{h^2} \begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots \\ & & & 1 & -2 & 1 \\ & & & & 1 & -2 \end{bmatrix} \begin{bmatrix} U_{0,n} \\ U_{1,n} \\ \vdots \\ U_{M,n} \\ U_{M+1,n} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix},$$

Note also that this equation only considers one particular n . In order to actually solve our entire system, we must include the nodes above and below as well. This can be done by considering a much larger matrix A and a much longer vector U . The latter being a stacked vector containing all $M + 1$ elements U_{0n}, \dots, U_{Mn} , followed by U_{01}, \dots, U_{M1} and so on. The matrix A is still the tridiagonal matrix with elements $[1, -4, 1]$, but now with size $(N \cdot M)^2$. Now, we are able to include the nodes U_{above} and U_{below} from the stencil. These two nodes are now – instead of being above and below U_{center} – to the sides, M nodes away. See figure ??.

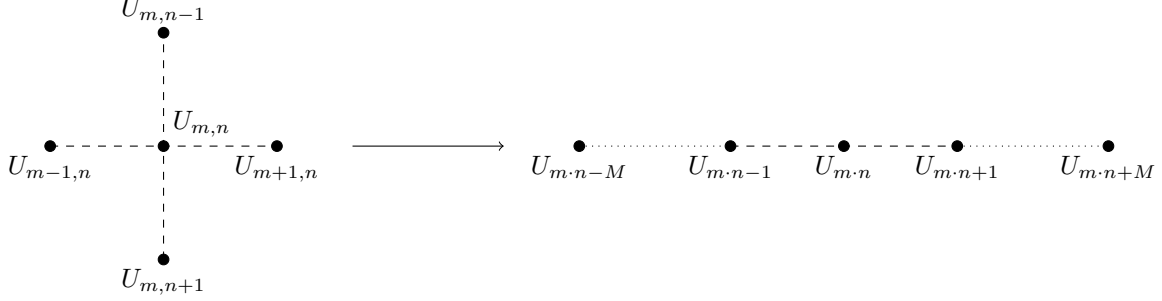


Figure 7: By flattening the five-point stencil, we can write the system of equations, which is now on the form $AU = 0$.

We thus write

$$AU = \begin{bmatrix} \frac{-2}{h^2} + \frac{-2}{k^2} & \frac{1}{h^2} & & \frac{1}{k^2} & & \\ \frac{1}{h^2} & \frac{-2}{h^2} + \frac{-2}{k^2} & \frac{1}{h^2} & & \frac{1}{k^2} & \\ & \frac{1}{h^2} & \frac{-2}{h^2} + \frac{-2}{k^2} & \frac{1}{h^2} & & \frac{1}{k^2} \\ & \ddots & \ddots & \ddots & & \\ \frac{1}{k^2} & & \frac{1}{h^2} & \frac{-2}{h^2} + \frac{-2}{k^2} & \frac{1}{h^2} & \\ & \frac{1}{k^2} & & \frac{1}{h^2} & \frac{-2}{h^2} + \frac{-2}{k^2} & \frac{1}{h^2} \\ & & \frac{1}{k^2} & \frac{-2}{h^2} + \frac{-2}{k^2} & \frac{1}{h^2} & \frac{1}{k^2} \end{bmatrix} \begin{bmatrix} U_0 \\ \vdots \\ U_m \\ \vdots \\ U_{N \cdot m} \\ \vdots \\ U_{N \cdot M} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \end{bmatrix} = 0,$$

which can be solved, given initial conditions on the boundary of Ω .

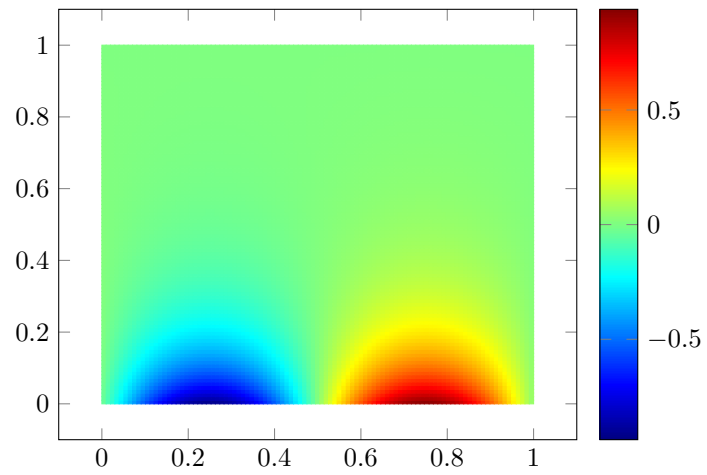


Figure 8: The numerically computed solution to the Laplace equation.

5 Linearized Korteweg-De Vries equation in one dimension

In this section, we will study the one-dimensional linearized Korteweg-De Vries equation

$$\frac{\partial u}{\partial t} + \left(1 + \pi^2\right) \frac{\partial u}{\partial x} + \frac{\partial^3 u}{\partial x^3} = 0 \quad (t \geq 0) \quad (-L/2 \leq x \leq +L/2), \quad (12)$$

where the solution $u = u(x, t)$ is subject to periodic boundary conditions

$$u(x + L, t) = u(x, t).$$

5.1 Analytical solution

As the solution is periodic in space at every time t , it can be expressed as a Fourier series

$$u(x, t) = \sum_{n=-\infty}^{+\infty} c_n(t) \exp(ik_n x) \quad (13)$$

with wavenumbers $k_n = 2\pi n/L$ and time-dependent coefficients $c_n(t)$ that ensure spatial periodicity at all times. This can be derived formally by separation of variables. Inserting the Fourier series into equation (12) gives the condition

$$\sum_n \left(\dot{c}_n(t) + i \left(\left(1 + \pi^2\right) k_n - k_n^3 \right) c_n(t) \right) \exp(ik_n x) = 0$$

on the coefficients. Due to orthogonality of the Fourier basis functions $\exp(ik_n x)$, the sum can vanish only if all prefactors vanish separately. This gives a first-order differential equation for each coefficient with the solution

$$c_n(t) = c_n(0) \exp \left(-i \left(\left(1 + \pi^2\right) k_n - k_n^3 \right) t \right). \quad (14)$$

5.2 Numerical solution method

To find a numerical solution $U_m^n = U(x_m, t_n) \approx u(x_m, t_n) = u_m^n$ of the Korteweg-De Vries equation, we will discretize it with central differences in space and integrate over time with the Euler method and the Crank-Nicholson method. For the first order spatial derivative, we use the central difference

$$\frac{\partial u_m^n}{\partial x} \approx \frac{\delta u_m^n}{2\Delta x} = \frac{u_{m+1}^n - u_{m-1}^n}{2\Delta x}.$$

We repeat the same finite difference three times to approximate the third order spatial derivative as

$$\frac{\partial^3 u_m^n}{\partial x^3} \approx \frac{\delta^3 u_m^n}{(2\Delta x)^3} = \frac{u_{m+3}^n - 3u_{m+1}^n + 3u_{m-1}^n - u_{m-3}^n}{8\Delta x^3}.$$

Inserting these approximations into equation (12), we get the intermediate result

$$\frac{\partial u_m^n}{\partial t} \approx F(u^n) = - \left(1 + \pi^2\right) \frac{u_{m+1}^n - u_{m-1}^n}{2\Delta x} - \frac{u_{m+3}^n - 3u_{m+1}^n + 3u_{m-1}^n - u_{m-3}^n}{8\Delta x^3}.$$

For later convenience, we write the Euler method and Crank-Nicholson method collectively with the θ -method. This gives the final system of difference equations for the numerical solution

$$\frac{U_m^{n+1} - U_m^n}{\Delta t} = (1 - \theta)F(U^n) + \theta F(U^{n+1}), \quad (15)$$

where the Euler method or the Crank-Nicholson method is obtained by inserting $\theta = 0$ or $\theta = 1/2$, respectively. In matrix form, the system can be written

$$(I - \theta \Delta t A) U^{n+1} = (I - (1 - \theta) \Delta t A) U^n, \quad (16)$$

where $U^n = [U_0^n \ \dots \ U_{M-1}^n]^T$ and $A =$

$$\frac{-1}{2\Delta x} \begin{bmatrix} 0 & +1 & & & & & -1 \\ -1 & 0 & +1 & & & & \\ & -1 & 0 & +1 & & & \\ & & -1 & 0 & +1 & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & -1 & 0 & +1 \\ & & & & & -1 & 0 & +1 \\ & & & & & & -1 & 0 \\ +1 & & & & & & & -1 \end{bmatrix} - \frac{1 + \pi^2}{\Delta x^3} \begin{bmatrix} 0 & -3 & 0 & +1 & & & -1 & 0 & +3 \\ +3 & 0 & -3 & 0 & +1 & & & -1 & 0 \\ 0 & +3 & 0 & -3 & 0 & +1 & & & -1 \\ -1 & 0 & +3 & 0 & -3 & 0 & +1 & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & -1 & 0 & +3 & 0 & -3 & 0 & +1 \\ +1 & & & -1 & 0 & +3 & 0 & -3 & 0 \\ 0 & +1 & & & -1 & 0 & +3 & 0 & -3 \\ -3 & 0 & +1 & & & -1 & 0 & +3 & 0 \end{bmatrix}.$$

where we have imposed periodic boundary conditions $U_m^n = U_{m+M}^n$ by simply wrapping the spatial derivative stencils around the matrix.

We then solve the system by preparing U^0 from the initial condition $u(x, 0)$ and solve equation (16) repeatedly to step forward in time. Note that with the constant time step Δt , all matrices in equation (16) are constant in time, and the process of solving the system many times can be accelerated by for example LU-factorizing the matrix on the left side.

Next, we test our numerical solution on the problem defined by the initial condition $u(x, 0) = \sin(\pi x)$ on $x \in [-1, +1]$ with $L = 2$. The Fourier series of the analytical solution then has nonzero coefficients $c_{\pm 1}(0) = \pm 1/2i$ and wavenumbers $k_{\pm 1} = \pm \pi$, which give rise to the analytical solution $u(x, t) = \sin(\pi(x - t))$ when inserted into equation (13). As shown in figure 9, the solution represents a sine wave traveling with velocity 1 to the right.

In figure 10, we compare snapshots of the numerical solution at $t = 1$ from the Euler method and the Crank-Nicholson method. Note that the Crank-Nicholson method approaches the exact solution with as little as $N = 10$ time steps and a few hundred spatial grid points M , while the Euler method produces garbage with

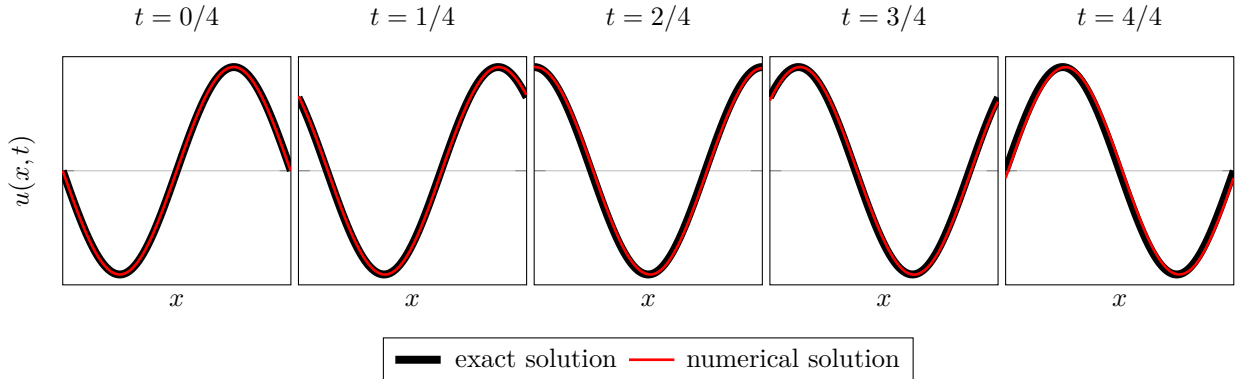


Figure 9: Comparison between the time evolution of the exact solution $u(x, t) = \sin(\pi(x - t))$ and the numerical solution from the Crank-Nicholson method with $\Delta x = 1/799$ and $\Delta t = 1/99$.

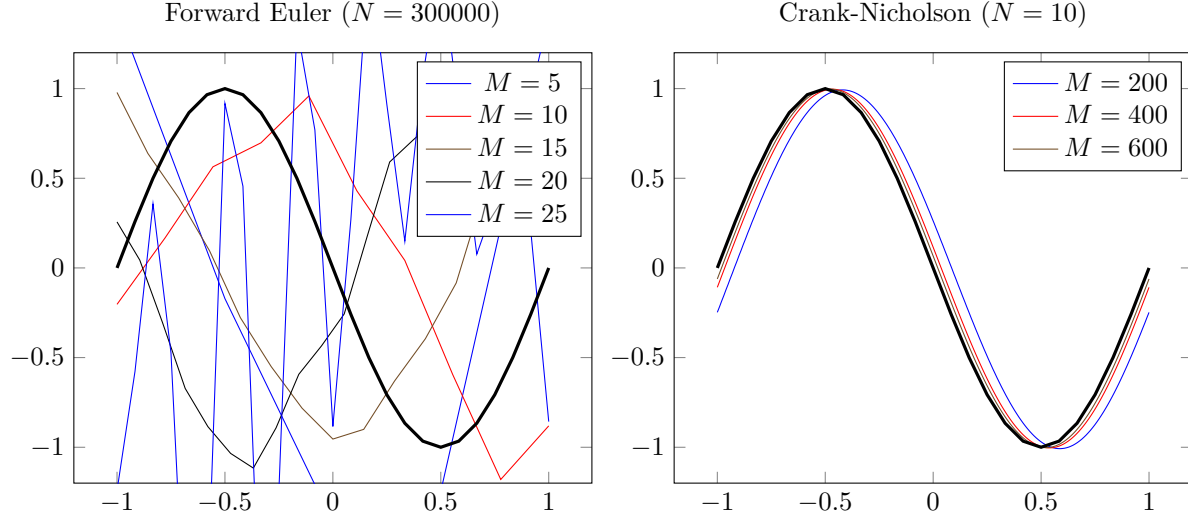


Figure 10

as many as $N = 300000$ time steps and becomes less stable as the number of spatial grid points increases. The convergence plot of the error at $t = 1$ in figure 11 supports our suspicions, showing the stable second order (first order???) nature of the Crank-Nicholson method and the instability of the Euler method.

5.3 Stability analysis

Motivated by the examples of the Euler method and the Crank-Nicholson method, we perform a Von Neumann analysis of their stability. Just like the exact solution, the numerical solution is subject to periodic boundary conditions in space and can therefore be expanded in a Fourier series

$$U_m^n = U(x_m, t_n) = \sum_l C_l^n \exp(ik_l x_m). \quad (17)$$

Consider now a single Fourier mode $C_l^n \exp(ik_l x_m)$ in this series. Inserting it into equation (15), dividing by $\exp(ik_l x_m)$ and expanding exponentials using Euler's identity gives

$$\frac{C_l^{n+1} - C_l^n}{\Delta t} = i \left((1 - \theta) C_l^n + \theta C_l^{n+1} \right) f(k_l), \quad \text{where } f(k_l) = \left(- \left(1 + \pi^2 \right) \frac{\sin(k_l h)}{h} - \frac{\sin^3(k_l h)}{h^3} \right).$$

Now look at the amplification factor $G_l = C_l^{n+1}/C_l^n$ of Fourier mode l over one time step. With $\theta = 1/2$, the Crank-Nicholson method gives

$$G_l = \frac{1 + i\Delta t f(x)/2}{1 - i\Delta t f(x)/2} \implies |G_l| = 1. \quad (18)$$

The amplitude of all Fourier modes is thus preserved over time independently of Δt and Δx , and we say the Crank-Nicholson method is **unconditionally stable**.

The Euler method has $\theta = 0$ and gives

$$G_l = 1 + i\Delta t f(x) \implies |G_l| = \sqrt{1 + \Delta t^2 f(k_l)^2}. \quad (19)$$

Since $|\sin(k_l h)| \leq 1$ for all k_l , we can bound $f(k_l)$ by

$$|f(k_l)| \leq \frac{(1 + \pi^2)}{\Delta x} + \frac{1}{\Delta x^3} = \frac{1}{\Delta x^3} \left((1 + \pi^2) \Delta x^2 + 1 \right) \leq \frac{1}{\Delta x^3} \left((1 + \pi^2) L^2 + 1 \right).$$

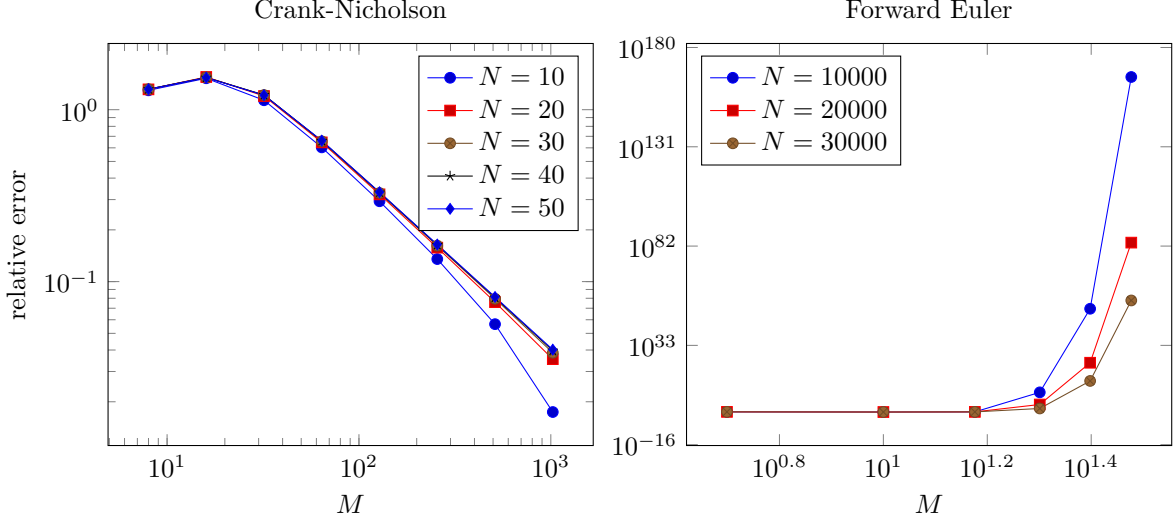


Figure 11: Convergence plot at $t = 1$, comparison between Crank-Nicholson and Forward Euler

Then $|G_l| = \sqrt{1 + O(\Delta t^2 / \Delta x^6)}$. The Von Neumann stability criterion $|G_l| \leq 1 + O(\Delta t)$ [1] is attained only with $\Delta t \leq O(\Delta x^6)$, corresponding to *extremely small* time steps. Thus, while the Euler method in theory is **conditionally stable**, it is unstable for all practical time steps. The Crank-Nicholson method is far superior, as it remains stable while allowing much coarser spatial resolution and larger time steps.

5.4 Time evolution of norm

The stability of the finite difference methods can be even better illustrated by investigating the time evolution of the L_2 -norm of the solution. To this end, we will first show that the L_2 -norm of the analytical solution is preserved over time. Then we will show the time evolution of the L_2 -norm of numerical solutions.

The L_2 -norm of the analytical solution is defined as

$$\|u(x, t)\|_2 = \left(\frac{1}{2} \int_{-L/2}^{+L/2} |u(x, t)|^2 dx \right)^{1/2}.$$

To understand why it is constant in time, insert the Fourier expansion equation (13) for $u(x, t)$ to get

$$\int_{-1}^{+1} dx |u(x, t)|^2 = \sum_{m,n} c_m(t) c_n^*(t) \underbrace{\int_{-L/2}^{+L/2} \exp(i(k_m - k_n)x) dx}_{L\delta_{mn}} = L \sum_n |c_n(t)|^2.$$

Since the Fourier coefficients (14) have constant magnitude $|c_n(t)| = |c_n(0)|$, the L_2 -norm is the same at any time t . Note that it is the *odd* number of spatial derivatives in equation (12) that gives the exponential in equation (14) a purely imaginary argument, and preserves the amplitude of each Fourier mode.

We now investigate the norm of the numerical solution with the initial gaussian $u(x, 0) = \exp(-x^2/0.1)$. The time evolution illustrated in figure 12 shows how multiple Fourier modes are activated. In figure 13, we show how the norm of the numerical solution evolves over time. The Euler method diverges even with tiny time steps, reflecting the amplification factor $G_l > 1$ found in equation (19). In contrast, the Crank-Nicholson method is always stable and preserves the norm of the solution, reflecting the amplification factor $G_l = 1$ found in equation (18).

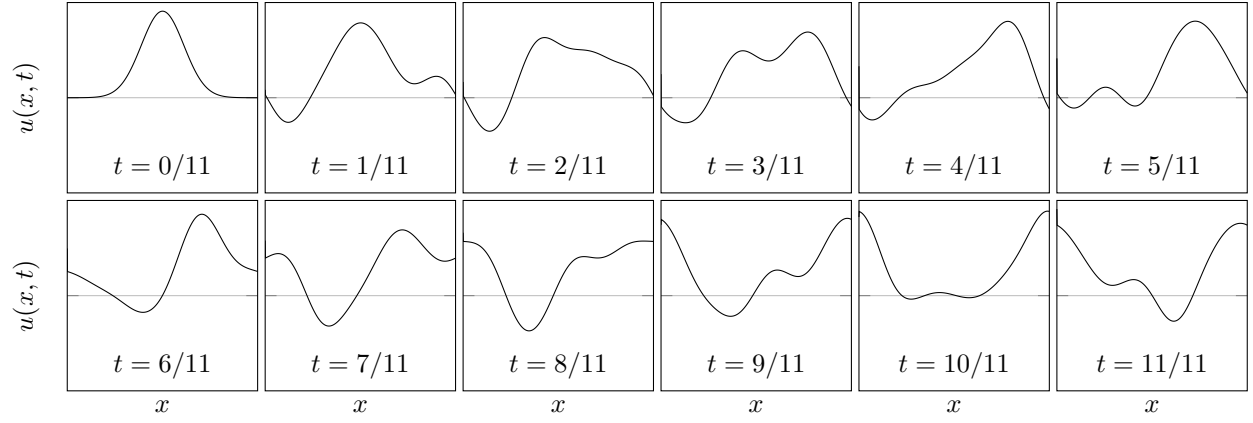


Figure 12: Time evolution of a initial gaussian $u(x, 0) = \exp(-x^2/0.1)$ computed from the Crank-Nicholson method on a grid with $M = 800$ points in space and $N = 100$ points in time.

The stability and norm preservation of the Crank-Nicholson method makes it the method of choice for problems like this, where the analytical solution is known to have the same property.

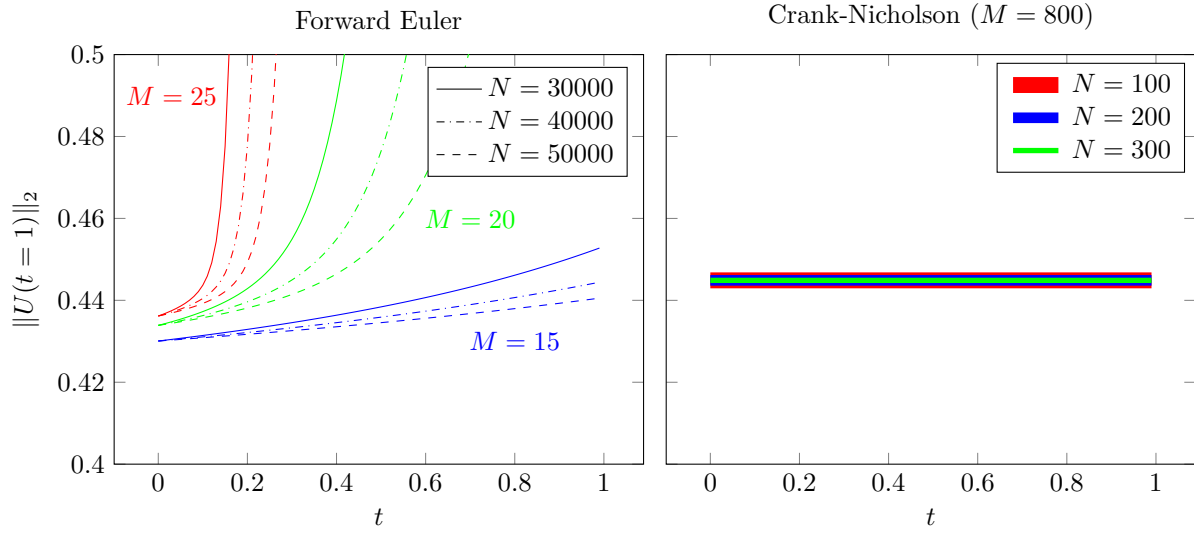


Figure 13: Time evolution of the discrete L_2 -norm of the initial gaussian $u(x, 0) = \exp(-x^2/0.1)$ computed from the Euler and the Crank-Nicholson method, on different grids.

6 Poisson equation in one dimension finite element method

In this section, we will again solve the Poisson equation

$$-\frac{\partial^2 u}{\partial x^2} = f(x), \quad u(a) = \alpha, \quad u(b) = \beta, \quad (a \leq x \leq b) \quad (20)$$

subject to Dirichlet conditions, but this time using finite elements instead of finite differences.

6.1 Analytical solution

The solution to the Poisson equation is the same as in 2, but with $f(x) \rightarrow -f(x)$, so that

$$u(x) = C_1 + C_2 x - \int^x dx' \int^{x'} dx'' f(x''). \quad (21)$$

6.2 Weak formulation for the exact solution

To derive a finite element method, we first split the solution into two terms

$$u(x) = \hat{u}(x) + r(x), \quad \text{with} \quad \hat{u}(a) = \hat{u}(b) = 0 \quad \text{and} \quad r(x) = \alpha \frac{x-b}{a-b} + \beta \frac{x-a}{b-a}. \quad (22)$$

Note that $u''(x) = \hat{u}''(x)$ and $r(a) = \alpha$ and $r(b) = \beta$. The purpose of this splitting is that \hat{u} solves 20 with homogenous Dirichlet boundary conditions, while $r(x)$ **lifts** the values at the boundaries to satisfy the inhomogenous boundary conditions.

Now insert 22 into equation (20), multiply it by a **trial function** $v(x)$ and integrate both sides from a to b . We let $v(a) = v(b) = 0$ and use integration by parts on the left, dropping the boundary term term $-[u'(x)v(x)]_a^b$. This gives the **weak formulation** of the problem:

$$\text{Find } \hat{u}(x) \text{ such that} \quad \int_a^b dx \hat{u}'(x)v'(x) = \int_a^b dx f(x)v(x) - \int_a^b dx r'(x)v'(x) \quad \text{for all } v(x). \quad (23)$$

The weak formulation 23 is equivalent to the original boundary value problem 20. Any $u(x)$ that solves 20 also solves 23, and it is possible to show that the converse is also true.

6.3 Weak formulation for the approximate solution

We have not made any approximations yet. The approximation lies in seeking a solution $U(x) \approx u(x)$ that belongs to a function space different from the one in which the exact solution $u(x)$ belongs. Here, we suppose $U(x)$ lies in the space of piecewise linear functions. We will then repeat the process above to derive a weak formulation for $U(x)$, similarly as for the exact solution.

To see how this works, we first divide the interval $[a, b]$ into the grid

$$a = x_0 < x_1 < \dots < x_M < x_{M+1} = b \quad (24)$$

and let $U(x)$ be piecewise linear in each **finite element** $[x_i, x_{i+1}]$. Similarly to $u(x)$, we split the approximate solution into

$$U(x) = \hat{U}(x) + R(x), \quad \text{with} \quad \hat{U}(a) = \hat{U}(b) = 0 \quad \text{and} \quad R(x) = \begin{cases} \alpha \frac{x_1 - x}{x_1 - a} & (a \leq x \leq x_1) \\ 0 & (x_1 \leq x \leq x_M) \\ \beta \frac{x - x_M}{b - x_M} & (x_M \leq x \leq b) \end{cases} \quad (25)$$

Now again insert 26 into 20, multiply by a trial function $V(x)$ that vanishes at a and b , integrate from a to b and drop a boundary term. This leads to the weak formulation for the approximate solution:

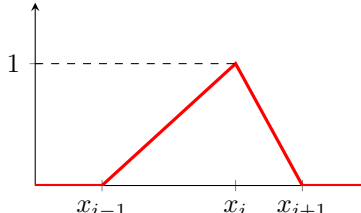
$$\text{Find } \hat{U}(x) \text{ such that } \int_a^b dx \hat{U}'(x) V'(x) = \int_a^b dx f(x) V(x) - \int_a^b dx R'(x) V'(x) \quad \text{for all } V(x). \quad (26)$$

6.4 Numerical solution

To obtain a matrix equation for approximate solution $U(x)$, the next step is to expand

$$\hat{U}(x) = \sum_{i=0}^{M+1} \hat{U}_i \varphi_i(x) \quad \text{and} \quad V(x) = \sum_{i=0}^{M+1} V_i \varphi_i(x) \quad (27)$$

in a basis for the approximate solution function space, namely the piecewise linear functions on the grid 24. The most natural basis for this space are the functions

$$\varphi_i(x) = \begin{cases} (x - x_{i-1})/(x_i - x_{i-1}) & \text{if } x_{i-1} \leq x \leq x_i \\ (x_{i+1} - x)/(x_{i+1} - x_i) & \text{if } x_i \leq x \leq x_{i+1} \\ 0 & \text{otherwise} \end{cases} \quad (28)$$


With this basis, the coefficients $\hat{U}_i = \hat{U}(x_i)$ are simply the values at the grid points, making it straightforward to plot the solution. Inserting this expansion into 26 gives

$$\begin{aligned} \sum_{i,j} \hat{U}_i V_j \int_a^b dx \varphi_i'(x) \varphi_j'(x) &= \sum_j V_j \int_a^b dx \varphi_j(x) f(x) \\ &\quad - a \sum_j V_j \int_a^b dx \varphi_0'(x) \varphi_j'(x) - b \sum_j V_j \int_a^b dx \varphi_{M+1}'(x) \varphi_j'(x). \end{aligned}$$

We can write this as the neat matrix equation $V^T A \hat{U} = V^T F$ by introducing

$$\hat{U} = [\hat{U}_1, \dots, \hat{U}_M]^T, \quad V = [V_1, \dots, V_M]^T, \quad A_{ij} = \int_a^b dx \varphi_i'(x) \varphi_j'(x) \quad \text{and} \quad F_j = \int_a^b dx \varphi_j(x) f(x).$$

for $0 \leq i, j \leq M+1$. Since this must hold for *any* $V(x)$ and thus V , we must have

$$A \hat{U} = F. \quad (29)$$

This is the matrix equation we will solve to find $U(x)$. After finding \hat{U} , we simply sum 27 and add $R(x)$ to find $U(x)$. Note that our particular choice of basis 28 gives the convenient property $U(x_i) = U_i$, so summing is not necessary in practice.

To solve 29, we must first calculate the so-called **stiffness matrix** A and the **load vector** F . The former involves only the known basis functions and gives nonzero entries

$$\begin{aligned} A_{00} &= \frac{1}{x_1 - x_0} & A_{M+1M+1} &= \frac{1}{x_{M+1} - x_M} \\ A_{ii} &= \frac{1}{x_i - x_{i-1}} + \frac{1}{x_{i+1} - x_i} & A_{ii+1} = A_{i+1i} &= \frac{1}{x_{i+1} - x_i}. \end{aligned}$$

The latter involves integrals over an arbitrary source function $f(x)$ times the basis functions $\varphi_j(x)$. This integral must be approximated numerically and should be split from x_{i-1} to x_i and x_i to x_{i+1} to properly handle the spike in $\varphi_j(x)$ at x_j . (TODO: comment on numerical integration procedure, Gaussian integration)

We now impose $\hat{U}(a) = \hat{U}(b) = 0$ by removing the first and last entries in the matrix equation *after* calculating the entire $(M+2) \times (M+2)$ system described above. This gives an $M \times M$ equation. Then we construct U by appending α and β at the beginning and end of the M -vector \hat{U} .

6.4.1 Uniform refinement

We test our method on four problems, shown in figure 13, with uniform elements $x_i - x_{i-1} = (b-a)/(M+1)$.

The approximate solutions resembles the exact solution with few points. For the symmetric Gaussian problems, it is vital to choose an odd number of grid points to capture the spike in the center. In the final problem, the source function diverges at the left boundary, but the numerical integration is still able to find a good numerical solution.

In all but the first problem, errors distribute non-evenly across the elements. Computational resources are wasted by using many points in areas where the solution varies slowly. These resources would be better spent by increasing the grid resolution in the areas where the error is large. This is the motivation for turning to adaptive refinement and non-uniform grids.

6.4.2 Adaptive refinement

Motivated by the uneven error distribution from using uniform elements, we will now do adaptive refinement, similarly to what we did in section 1.3. We start with a uniform grid and successively split those elements on which the error is largest. Contrary to what we did in section 1.3, we will not split only *one* element between each iteration of the numerical solution, but split *all* elements on which the error is greater than some reference error. This leaves us with less control over the number of elements, but in return we will see that the error strictly decreases in each iteration, eliminating the oscillating error in figure 3.

This time, we use two strategies that both involve the exact error:

1. **Average error strategy:** Split the interval $[x_m, x_{m+1}]$ with error

$$\|u(x) - U(x)\|_2 > 0.99 \frac{\|u(x) - U(x)\|_2}{N},$$

where N is the number of intervals. The safety factor $0.99 \approx 1$ ensures that intervals are split also when all errors are equal (up to machine precision), so the procedure does not halt unexpectedly.

2. **Maximum error strategy:** Split the interval $[x_m, x_{m+1}]$ with error

$$\|u(x) - U(x)\|_2 > 0.70 \max \|u(x) - U(x)\|_2,$$

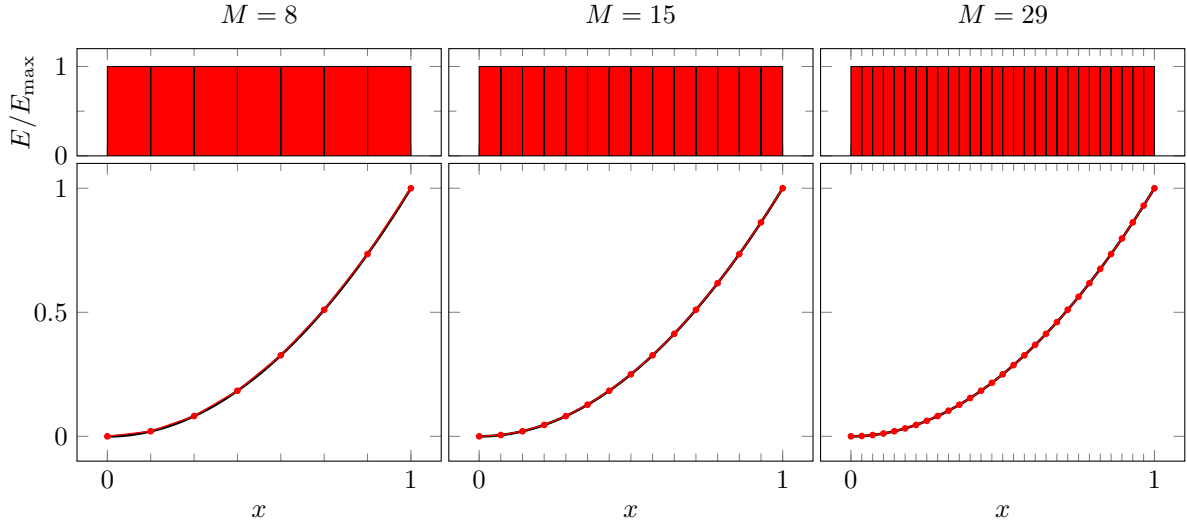
where N is the number of intervals.

In figure 13, we show how the errors distribute on the same four problems as in figure 13 using the average error strategy.

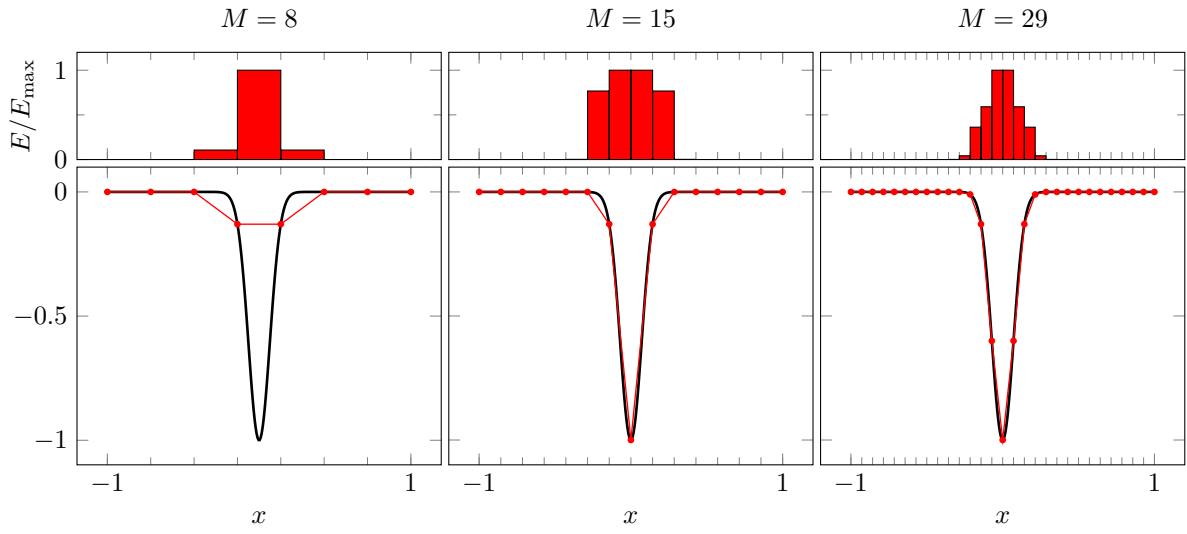
Observe how only elements with large error are refined, while others are left untouched. In the symmetric Gaussian problems, the refinement ensure that the middle element is split immediately if we do not start with a grid point at the peak. In the final problem, we see that it is almost only elements close to the left boundary where the source diverges that needs to be refined.

As discussed above, we see that the errors in the first problem distribute evenly on the initial uniform grid. This shows the importance of the safety factor 0.99 in the average error strategy. Without it, precision issues would make some elements skip the refinement criterion.

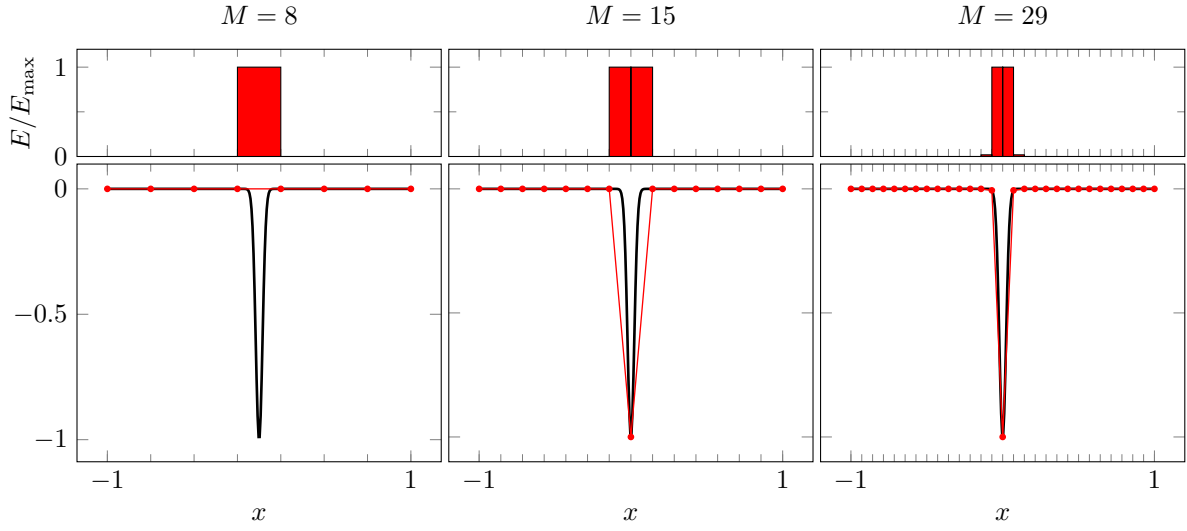
$$f(x) = -2$$



$$f(x) = (40000x^2 - 200)\exp(-100x^2)$$



$$f(x) = (4000000x^2 - 2000)\exp(-1000x^2)$$



$$f(x) = 2x^{-4/3}/9$$

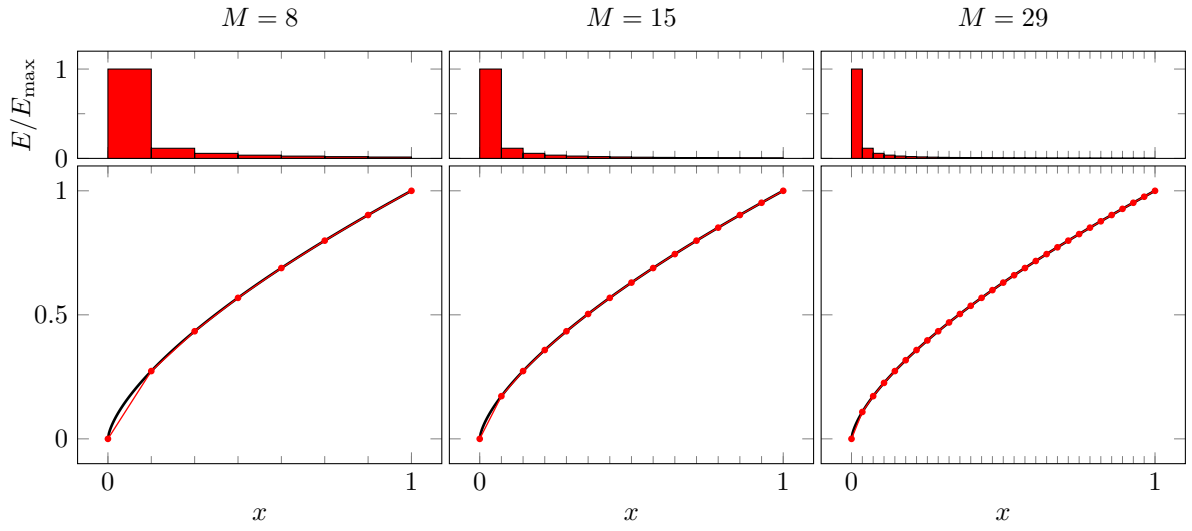


Figure 13: Uniform refinement

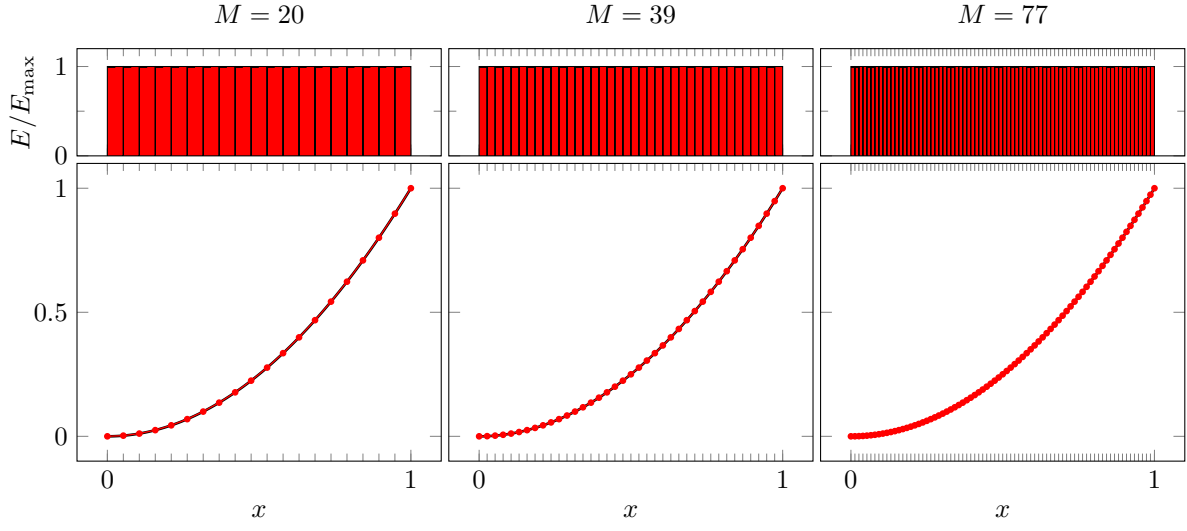
6.4.3 Comparison of convergence

Finally, in figure 14, we compare the convergence of uniform and adaptive refinement strategies.

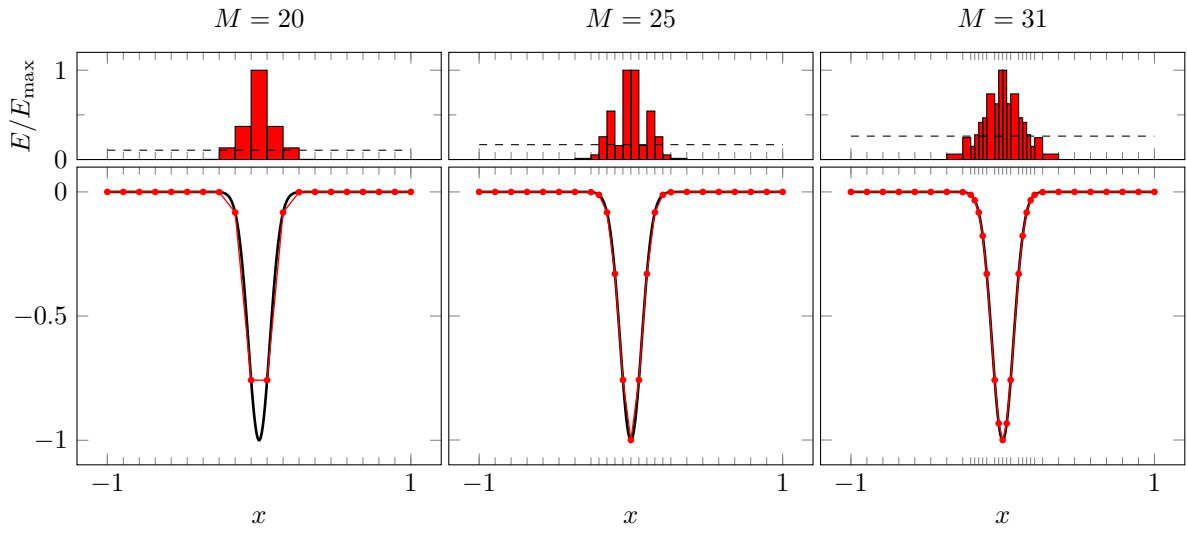
In all problems, adaptive refinement yields greater or equal accuracy for a given number of elements compared to uniform refinement. This is in contrast to what was the case for the finite difference method in figure 3, where adaptive refinement gave errors only comparable and usually larger than those from uniform refinement. It is only in the first problem that all strategies behave identically, as the errors here distribute evenly across the elements.

By splitting multiple intervals between each iteration of the numerical solution, we have eliminated the oscillating error pattern in figure 3. Now the error strictly decreases between each refinement of the grid. This suggests that the oscillating pattern is due to refinements where intervals with large error are present even after refining the element with greatest error. For example, it would be a bad idea to refine only *one* element in the first problem in figure 13, where errors are even across the elements.

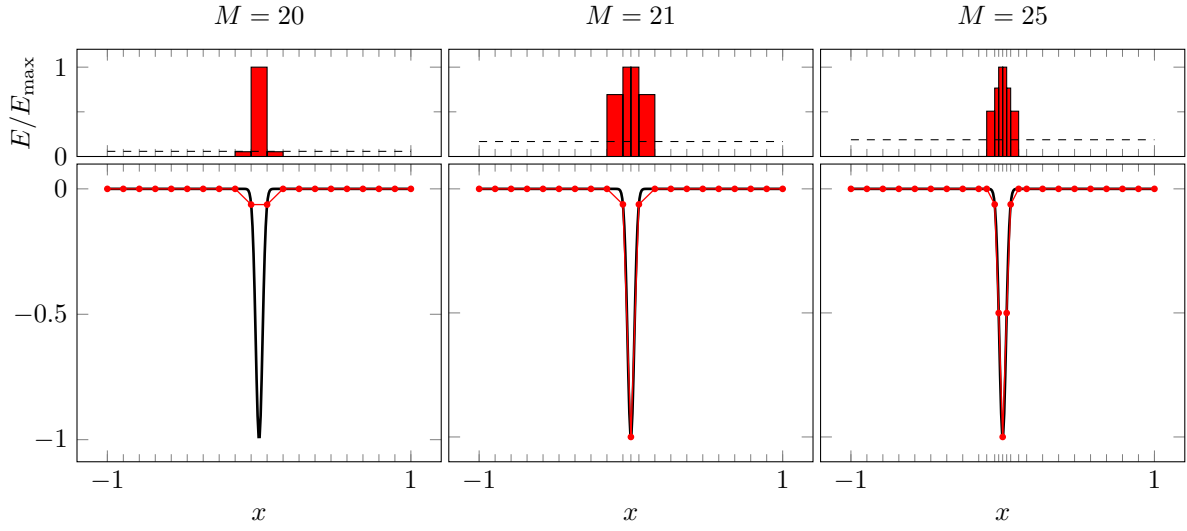
$$f(x) = -2$$



$$f(x) = (40000x^2 - 200) \exp(-100x^2)$$



$$f(x) = (4000000x^2 - 2000)\exp(-1000x^2)$$



$$f(x) = 2x^{-4/3}/9$$

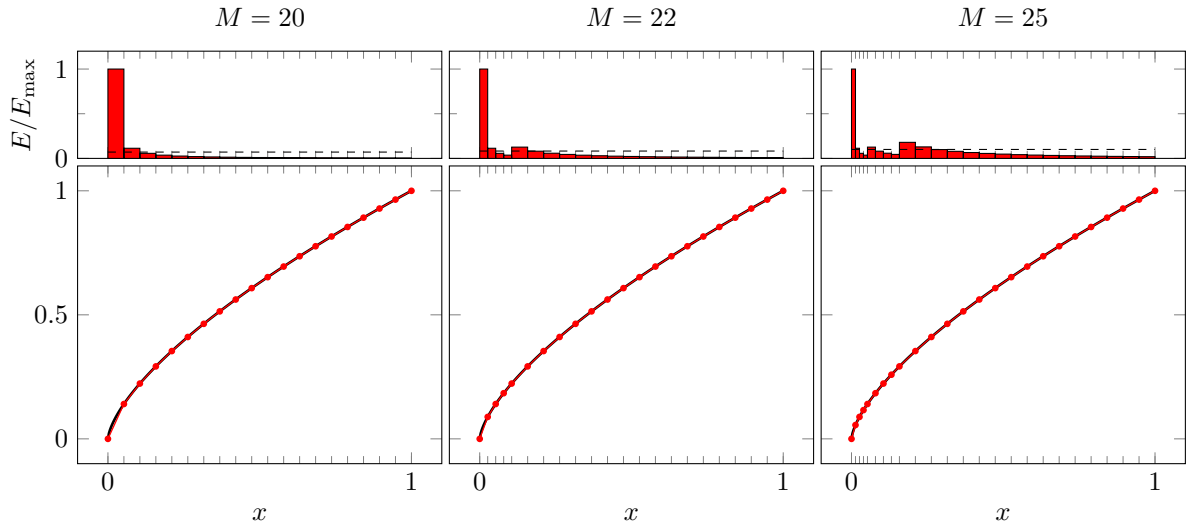


Figure 13: Adaptive refinement, average strategy

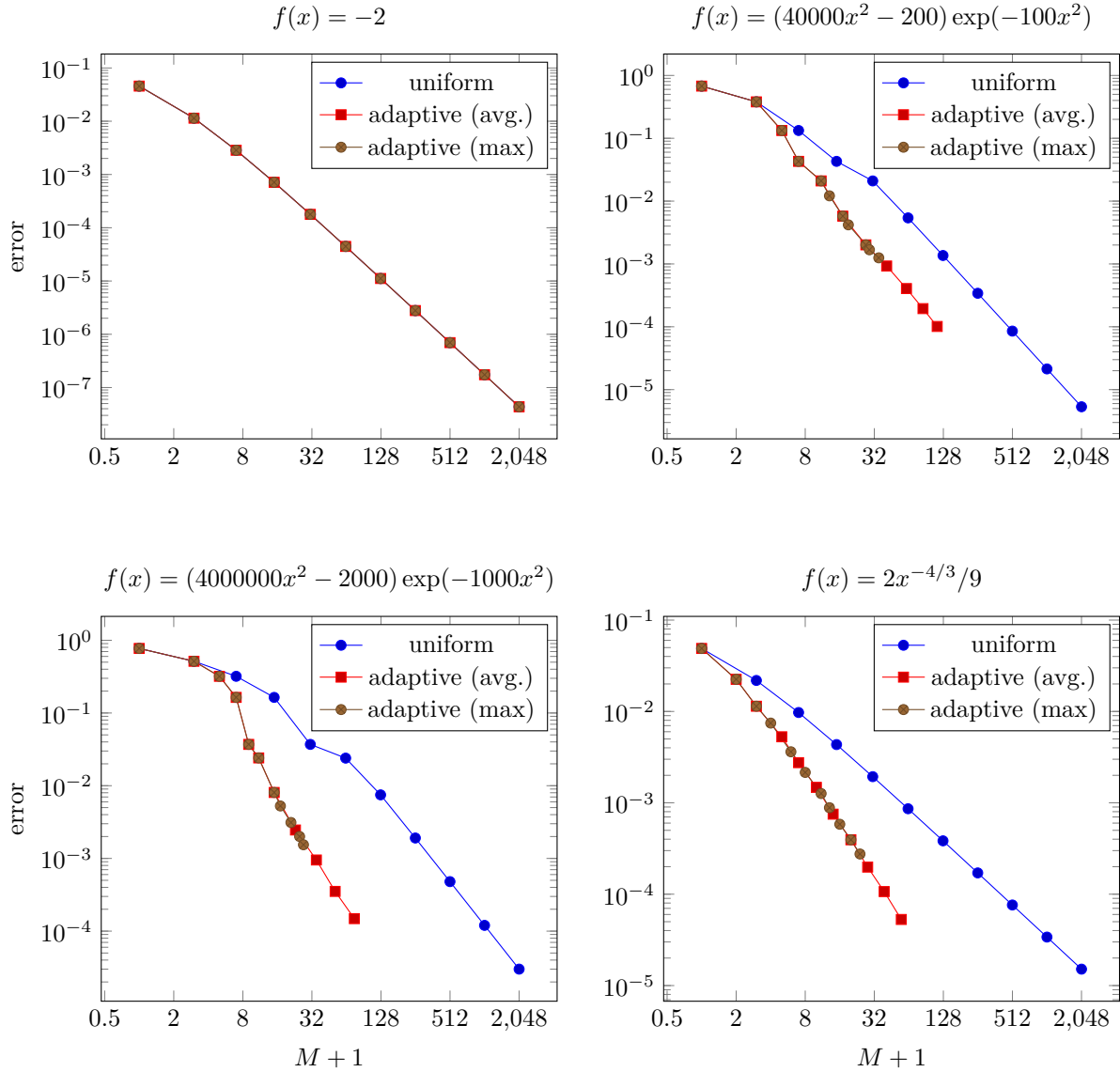


Figure 14: Convergence plot, comparison of three strategies

7 Biharmonic equation

Consider the inhomogeneous Biharmonic equation with clamped boundary conditions on the unit square $\Omega = [0, 1]^2$:

$$\nabla^4 u = f, \quad (x, y) \in \Omega, \quad (30a)$$

$$u = 0, \nabla^2 u = 0, \quad (x, y) \in \partial\Omega. \quad (30b)$$

Firstly, note that the functions

$$\sin(m\pi x) \sin(n\pi y), \quad m, n = 1, 2, \dots$$

represent a complete basis for functions on $\Omega = [0, 1]^2$ with a boundary value of 0. (Which is easily seen from the fact that the Fourier series consist of such a set, and that for the given boundary condition, the coefficients of terms involving cosines and constant functions must be zero) (TODO: comment on how to obtain this with separation of variables?) Thus, for any function u that has $u = 0$ on (Ω) , we must be able to write

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

Inserting the expression into the Biharmonic equation, we get, after acting on the expression with the biharmonic operator ∇^4

$$\nabla^4 u = \sum_{m=1}^{\infty} \sum_{n=1}^{\infty} F_{mn} \nabla^4 \sin(m\pi x) \sin(n\pi y) \quad (31)$$

$$= \sum_{m=1}^{\infty} \sum_{n=1}^{\infty} F_{mn} \left((m\pi)^4 + (n\pi)^4 + 2n^2 m^2 \pi^4 \right) \sin(m\pi x) \sin(n\pi y) \quad (32)$$

$$= \sum_{m=1}^{\infty} \sum_{n=1}^{\infty} F_{mn} \left((m\pi)^2 + (n\pi)^2 \right)^2 \sin(m\pi x) \sin(n\pi y) \quad (33)$$

$$= f. \quad (34)$$

Thus, with

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

we must have that

$$F_{mn} = \frac{G_{mn}}{\left((m\pi)^2 + (n\pi)^2 \right)^2}$$

(TODO: comment on orthogonal sine functions?) Observe that choosing f such that the series is finite, ie. only a finite number of G_{mn} are non-zero, the expression collapses into a finite expression.

We may transform (30) into a system of Poisson equation by introducing $g = \nabla^2 u$,

$$\nabla^2 g = f, \quad (35)$$

$$\nabla^2 u = g, \quad (36)$$

$$g = u = 0, \quad \text{on } \partial\Omega. \quad (37)$$

In solving the Poisson equation $\nabla^2 u = f$ will consider here two discretization schemes, the five point stencil and nine point stencil, which we will denote ∇_5^2 and ∇_9^2 .

Written in stencil diagrams, the five point stencil is given as

$$u \begin{pmatrix} & \textcircled{1} & \\ \textcircled{1} & -4 & \textcircled{1} \\ & \textcircled{1} & \end{pmatrix} = h^2 f$$

while the nine point stencil is given by

$$u \left(\begin{pmatrix} & \textcircled{\frac{1}{6}} & \textcircled{\frac{1}{6}} & \\ \textcircled{\frac{1}{6}} & & \textcircled{\frac{1}{6}} & \\ & \textcircled{\frac{1}{6}} & \textcircled{\frac{1}{6}} & \end{pmatrix} + \begin{pmatrix} & \textcircled{\frac{2}{3}} & \\ \textcircled{\frac{2}{3}} & -\frac{10}{3} & \textcircled{\frac{2}{3}} \\ & \textcircled{\frac{2}{3}} & \end{pmatrix} \right) = h^2 f \begin{pmatrix} & \textcircled{\frac{1}{12}} & \\ \textcircled{\frac{1}{12}} & \textcircled{\frac{2}{3}} & \textcircled{\frac{1}{12}} \\ & \textcircled{\frac{1}{12}} & \end{pmatrix}.$$

We will now describe the structure of the matrices involved in the problem, starting with the simpler five point stencil. We will use the same flattening for the 2D discrete function u as described in section 4.2. We recognize that in U neighbouring elements correspond to grid points that are left and right of each other, while the grid point up and down is the element N places before and after. Thus, writing out the five point stencil $K2D$, we get a toeplitz and symmetric matrix, where the main diagonal has -4 and the ± 1 and $\pm N$ off diagonals have 1 . We can compactly write this as

$$K2D = \begin{bmatrix} K & 0 & & \\ 0 & K & 0 & \\ 0 & 0 & K & \\ & & & \ddots \end{bmatrix} + \begin{bmatrix} -2I & I & 0 & & \\ I & -2I & I & 0 & \\ 0 & I & -2I & I & 0 \dots \\ \vdots & & & \ddots & \end{bmatrix}.$$

where I is the identity matrix and K is the one dimensional central finite difference matrix of order 2,

$$K = \begin{bmatrix} -2 & 1 & & & \\ & 1 & -2 & 1 & \\ & & 1 & -2 & 1 \\ & & & \ddots & \\ & & & & 1 & -2 & 1 \\ & & & & & 1 & -2 \end{bmatrix}.$$

Using the Kronecker product, this may be written as

$$K2D = \text{kron}(I, K) + \text{kron}(K, I)$$

which is useful when working with sparse matrices.

We will split the nine point stencil matrix into two, according to the stencil diagram above. One matrix represents a five point stencil with weights altered to $-\frac{10}{3}, \frac{2}{3}$ instead of $-4, 1$, we denote this matrix by $K2D^{(9)}$. The other is the “X”-stencil taking care of the NE, NW, SE, and SW points, each with a weight of $\frac{1}{6}$, as shown in the diagram. We will denote this matrix by $\Sigma 2D$.

Let

$$\Sigma = \frac{1}{\sqrt{6}} \begin{bmatrix} 0 & 1 & & & \\ 1 & 0 & 1 & & \\ & 1 & 0 & 1 & \\ & & 1 & 0 & \ddots \\ & & & \ddots & \ddots \end{bmatrix} \quad (38)$$

be the TST matrix with ones on its off-diagonals and zero on the diagonal. Then we have $\Sigma 2D = \Sigma \otimes \Sigma$. The nine point stencil is then

$$\text{somename9stencil} = K2D^{(9)} + \Sigma 2D \quad (39)$$

The eigenvalues of the Kronecker product $A \otimes B$ is the product of the eigenvalues of A and B . The eigenvalue of the Kronecker sum $A \oplus B = A \otimes I + I \otimes B$ is the sum of eigenvalues of A and B . In general the eigenvalues of a TST matrix are

$$\lambda_k = a + 2b \cos\left(\frac{k\pi}{N+1}\right)$$

where a is the main diagonal and b is the off diagonal. Thus, the eigenvalues of $K2D$ are

$$\left(-2 + 2 \cos\left(\frac{k\pi}{N+1}\right)\right) + \left(-2 + 2 \cos\left(\frac{l\pi}{N+1}\right)\right).$$

The eigenvalues of $K2D^{(9)}$ are

$$\frac{1}{3} \left(-10 + 4 \cos\left(\frac{k\pi}{N+1}\right)\right) + \frac{1}{3} \left(-10 + 4 \cos\left(\frac{l\pi}{N+1}\right)\right)$$

and the eigenvalues of $\Sigma \otimes \Sigma$ are

$$\frac{4}{6} \cos\left(\frac{k\pi}{N+1}\right) \cos\left(\frac{l\pi}{N+1}\right).$$

The eigenvalues of the nine point stencil are thus

$$-\frac{10}{3} + \frac{4}{3} \left(\cos\left(\frac{k\pi}{N+1}\right) + \cos\left(\frac{l\pi}{N+1}\right)\right) + \frac{4}{6} \cos\left(\frac{k\pi}{N+1}\right) \cos\left(\frac{l\pi}{N+1}\right). \quad (40)$$

7.1 Stability and order of the five point stencil

We will here prove that the five point stencil is of order 2, following the outline provided in the exercise.
TODO: stability?

Lemma 1. *If $\nabla_5^2 f \geq 0$ on Ω , then the maximum value of f is attained on $\partial\Omega$, that is*

$$\max(f)_\Omega \leq \max(f)_{\partial\Omega}.$$

Proof. Suppose the opposite is true,

$$\max(f)_\Omega > \max(f)_{\partial\Omega}.$$

Then, there is an internal grid point on which f attains it maximal value. We denote this point by x_0 , and label its neighbouring points $x_i, i = 1, 2, 3, 4$. Then

$$4f(x_0) = \sum_{i=1}^4 f(x_i) - \nabla_5^2 f(x_0) \leq \sum_{i=1}^4 f(x_i) \leq 4f(x_0),$$

where the last inequality is attained from the fact that x_0 is assumed to be the maximal value of f . As the RHS is equal to the LHS, the equality must hold throughout, and as $f(x_i) \leq f(x_0)$ we must have that all the neighbouring values have the same value as x_0 , $f(x_1) = f(x_2) = f(x_3) = f(x_4) = f(x_0)$. Applying this same argument to each of the neighbours, and then to their neighbours and so on, we ultimately reach the conclusion that all internal points have the same value, and that the same value is also attained on $\partial\Omega$, and we thus have a contradiction. \square

Lemma 2. *If v is a discrete function that equals zero on $\partial\Omega$, then*

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

Proof. Consider the function

$$\phi = \frac{1}{4}[(x - \frac{1}{2})^2 + (y - \frac{1}{2})^2],$$

It has the properties that $0 \leq \phi \leq \frac{1}{8}$ and $\nabla_5^2 \phi = 1$. Let $\|\nabla_5^2 v\|_\infty = M$. Then

$$\nabla_5^2(v + \phi M) = \nabla_5^2 v + M \geq 0.$$

By lemma 1 we have then that $\nabla_5^2(v + \phi M)$ attains its maximum value on $\partial\Omega$. Thus

$$\|v\|_\infty \leq \|v + \phi M\|_\infty \leq \max_{\partial\Omega}(v + \phi M) = \max_{\partial\Omega}(\phi M) = \frac{1}{8} \|\nabla_5^2 v\|_\infty.$$

\square

Theorem 1. *If $\nabla^2 u = f$ and $\nabla_5^2 v = f$, then*

$$\|u - v\|_\infty \leq Ch^4 |D^4 v|_\infty.$$

Proof. By lemma 2

$$\|u - v\|_\infty \leq \frac{1}{8} \|\nabla_5^2(u - v)\|_\infty.$$

By Taylor expanding $u(x + h, y)$ and $u(x - h, y)$ we get

$$\frac{1}{h^2} \delta_x^2 u = \partial_x^2 u + Ch^2 \partial_x^4,$$

and similarly for y . Therefore

$$\begin{aligned} \nabla_5^2 u &= \frac{1}{h^2} (\delta_x^2 + \delta_y^2) u \\ &= (\partial_x^2 + \partial_y^2 + C_x h^2 \partial_x^4 + C_y h^2 \partial_y^4) u \\ &= (\nabla^2 + C_x h^2 \partial_x^4 + C_y h^2 \partial_y^4) u \\ &\leq \nabla^2 u + Ch^2 \max(\partial_x^4 u, \partial_y^4 u) \\ &\leq \nabla^2 u + Ch^2 |D^4 u|_\infty. \end{aligned}$$

Here $|D^n u|_\infty$ is to be understood as the maximal value of all n -index derivatives of u . For example for D^2 this would be u_{xx}, u_{xy}, u_{yx} , and u_{yy} . \square

7.2 Stability and order of the nine point stencil

Our proof for the nine point stencil will follow the same structure as for the five point stencil.

Lemma 3. *If $\nabla_9^2 f \geq 0$ on Ω , then the maximum value of f is attained on $\partial\Omega$, that is*

$$\max(f)_\Omega \leq \max(f)_{\partial\Omega}.$$

Proof. Notice that the nine point stencil may be written as a sum of two five point stencils – one defined normally and the other defined on the “diagonal grid” of our grid. With this we mean the grid of grid constant $\sqrt{2}h$, consisting of every other point of the original grid – ie. we reduce our grid into the black and white subgrids, following the chess analogy. Specifically, we have $\nabla_9^2 = \frac{2}{3}\nabla_5^2 + \frac{1}{6}\tilde{\nabla}_5^2$, where tilde indicates the operator on the aforementioned subgrid. As $\nabla_9^2 f \geq 0$, at least one of the five point stencils must also be positive. Thus, by lemma 1 the proof is completed. \square

Lemma 4. *If v is a discrete function that equals zero on $\partial\Omega$,*

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

Proof. Fuck, this was harder.... \square

Theorem 2. *If $\nabla^2 u = f$ and $\nabla_9^2 v = f$, then*

$$\|u - v\|_\infty \leq Ch^2|D^6 v|_\infty.$$

Proof. By lemma 4

$$\|u - v\|_\infty \leq \frac{3}{40}\|\nabla_5^2(u - v)\|_\infty.$$

By Taylor expanding $u(x + h, y)$ and $u(x - h, y)$ we get

$$\frac{1}{h^2}\delta_x^2 u = \partial_x^2 u + Ch^2\partial_x^4,$$

and similarly for y . Therefore

$$\begin{aligned} \nabla_5^2 u &= \frac{1}{h^2}(\delta_x^2 + \delta_y^2)u \\ &= (\partial_x^2 + \partial_y^2 + C_x h^2 \partial_x^4 + C_y h^2 \partial_y^4)u \\ &= (\nabla^2 + C_x h^2 \partial_x^4 + C_y h^2 \partial_y^4)u \\ &\leq \nabla^2 u + Ch^2 \max(\partial_x^4 u, \partial_y^4 u) \\ &\leq \nabla^2 u + Ch^2|D^4 u|_\infty. \end{aligned}$$

Here $|D^n u|_\infty$ is to be understood as the maximal value of all n -index derivatives of u . For example for D^2 this would be u_{xx}, u_{xy}, u_{yx} , and u_{yy} . \square

7.3 The Fast Poisson Solver

We are to solve the equation

$$AU = F.$$

Assuming that the eigenvectors of A are a complete set, we may write

$$F = a_1 y_1 + a_2 y_2 + \dots,$$

where y_1, y_2, \dots are the eigenvectors of A . The solution U is the of course

$$U = \frac{a_1}{\lambda_1} y_1 + \frac{a_2}{\lambda_2} y_2 + \dots,$$

where λ_i is the eigenvalue corresponding to y_i . In general, however, this is not a viable way to solve the problem, as it requires knowing all the eigenvalues and eigenvectors, as well as finding the coefficients a_i . However, for the set of eigenvectors in this problem, which are sines, we have a very efficient algorithm for computing the coefficients, the discrete fast sine transform. We also have simple analytical expressions for the eigenvalues.

Written as matrix expressions

$$AU = F \tag{41}$$

$$S\Lambda SU = F \tag{42}$$

$$\Rightarrow \tag{43}$$

$$U = S\Lambda^{-1}SF, \tag{44}$$

where we used the fact that $S^{-1} = S$. Moreover, formulated more directly with regards to implementing the solver, we have that the solution is

$$U = IFST(FST(F)/\Lambda),$$

where $IFST, FST$ are the inverse and normal fast sine transform.

References

- [1] Brynjulf Owren: *TMA4212 Numerical solution of partial differential equations with finite difference methods* (2017) [<http://www.math.ntnu.no/emner/TMA4212/2020v/notes/master.pdf>]