

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 use 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 in general 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 allows us to use the same stencil also at x_0 , but we must 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 them in a smart way.

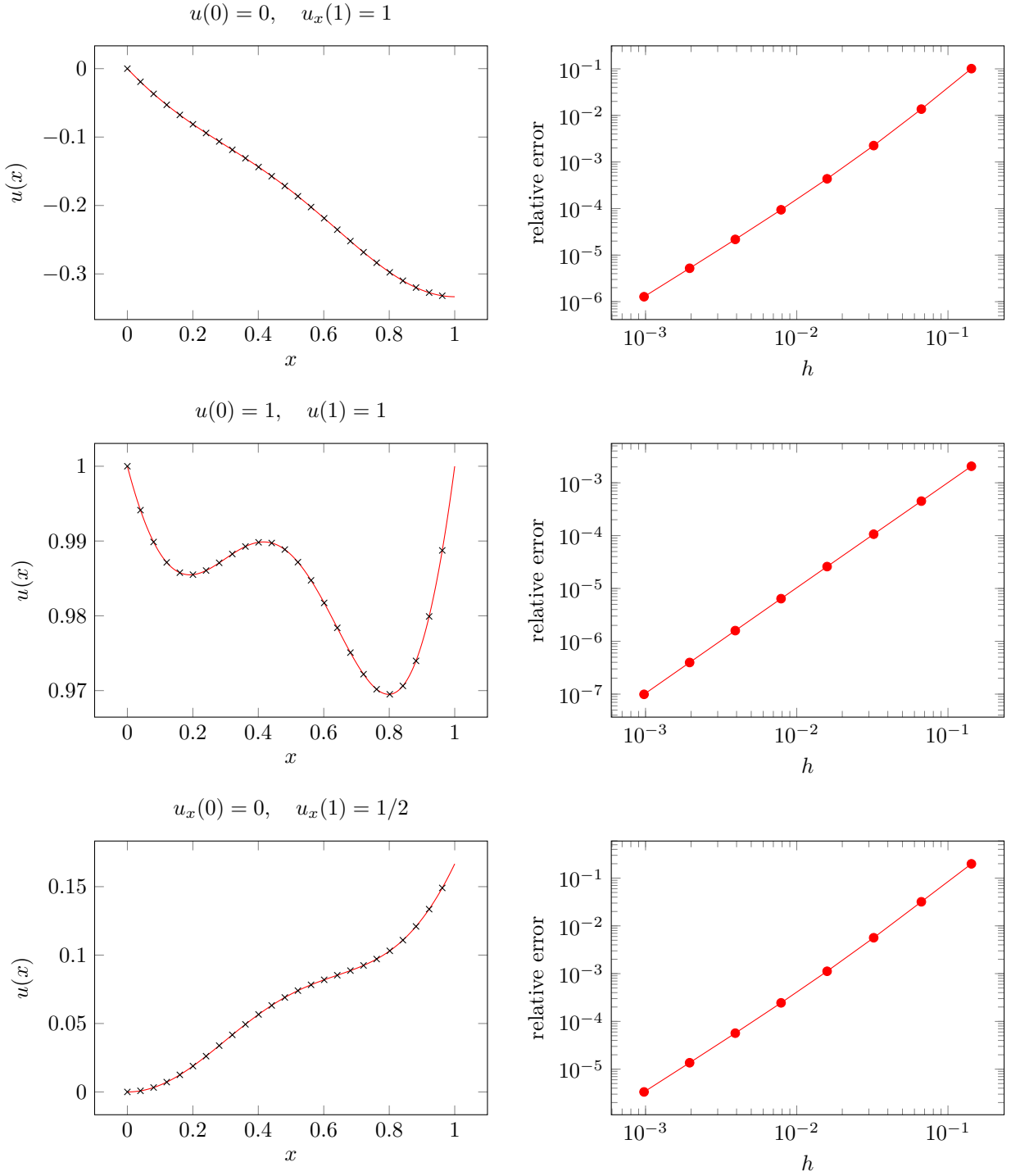


Figure 1: Analytical and numerical solutions (left) and convergence plots (right) for solutions to the Poisson equation subject to three different boundary conditions.

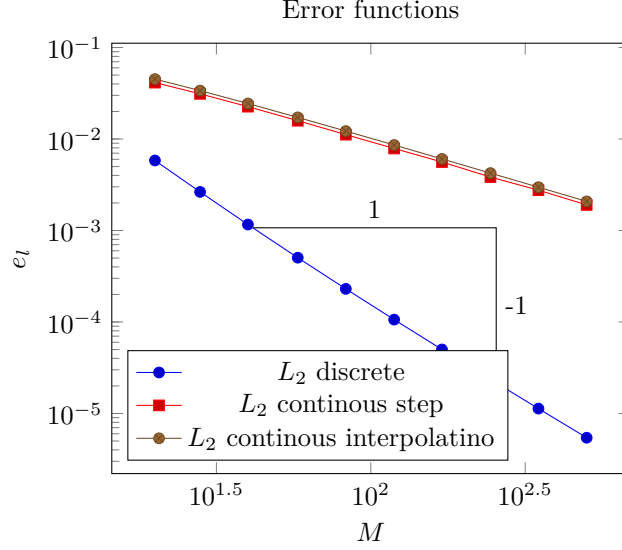


Figure 2: Nice fig

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

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, like with $x_0 = 0$ and $x_1 = 0$ only.
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

$$\left| \frac{2}{x_{m+1} - x_m} \left(\frac{u_{m+1} - u_{m+1/2}}{x_{m+1} - x_{m+1/2}} - \frac{u_{m+1/2} - u_m}{x_{m+1/2} - x_m} \right) - f(x_m) \right|, \quad \text{TODO fix interval indexes (m+1/2)}$$

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 .

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 given source function $f(x)$, which is the most realistic scenario.

In figure 3, we demonstrate how the grid and the numerical solution evolves as the grid $[0, 1/2, 1]$ with $M = 3$ is refined to $M = 25$. In figure 4, we show how the convergence of the three different AMR strategies compares to the second order UMR method. The error strategy and truncation error strategy makes AMR comparable to second order UMR, even though 4 has lower order! Moreover, the error-based AMR even produces a lower error than the UMR solution for select grids, and seems to do so in a periodic manner as the grid size M increases!

TODO discuss pros/cons of each strategy

TODO discuss “potential” of AMR

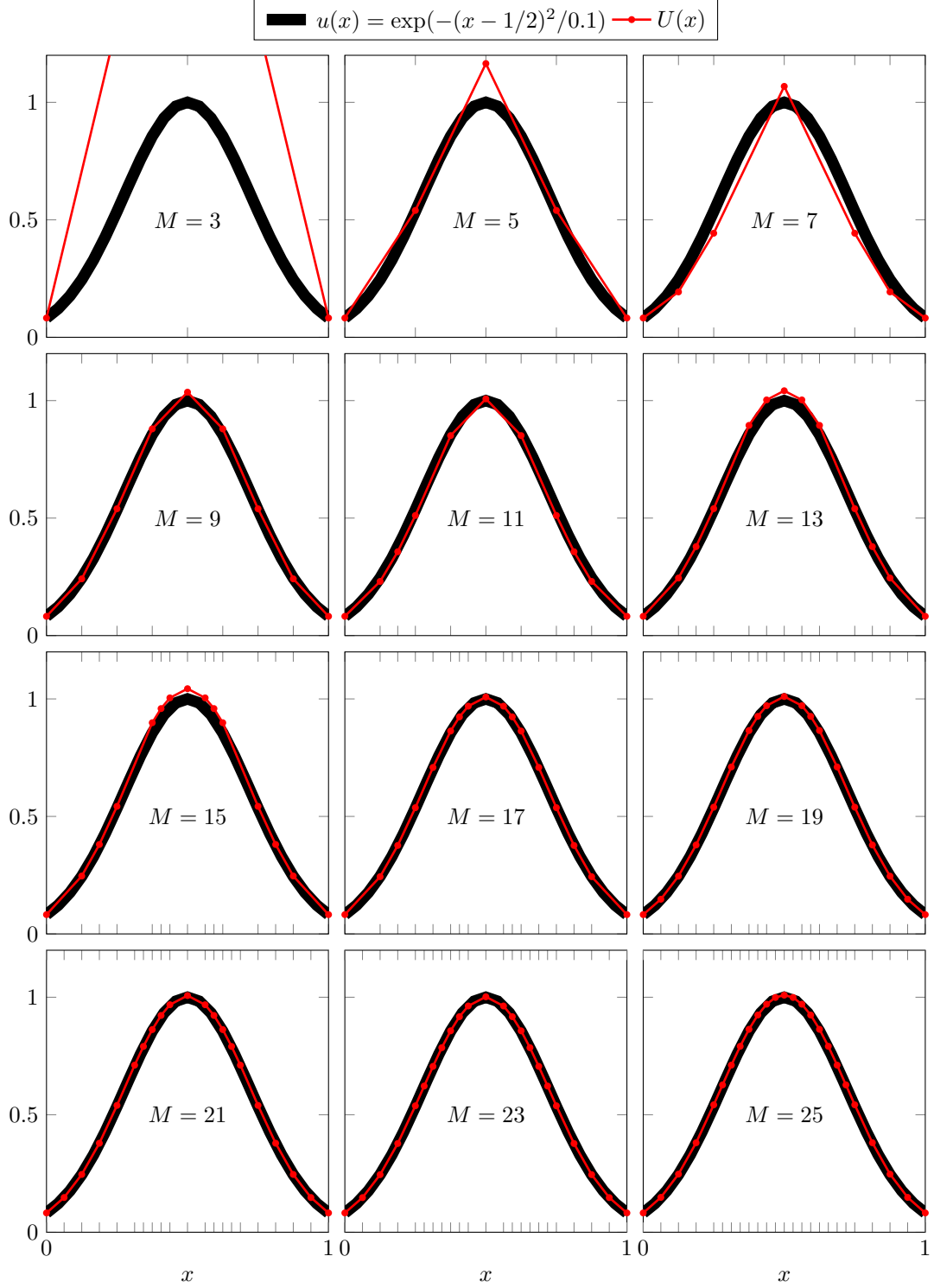


Figure 3: During adaptive mesh refinement (AMR) with the error strategy, the interval with the largest error $\int dx |u(x) - U(x)|$ is split in half. Here, $u(x) = \exp(-(x - 1/2)^2 / 0.1)$ is the symmetric solution to whichever Poisson equation has $f(x) = u_{xx}$ on $x \in [0, 1]$. Symmetry is imposed numerically by also adding the point $1 - x$ to the grid whenever a point $x \neq 1/2$ is added.

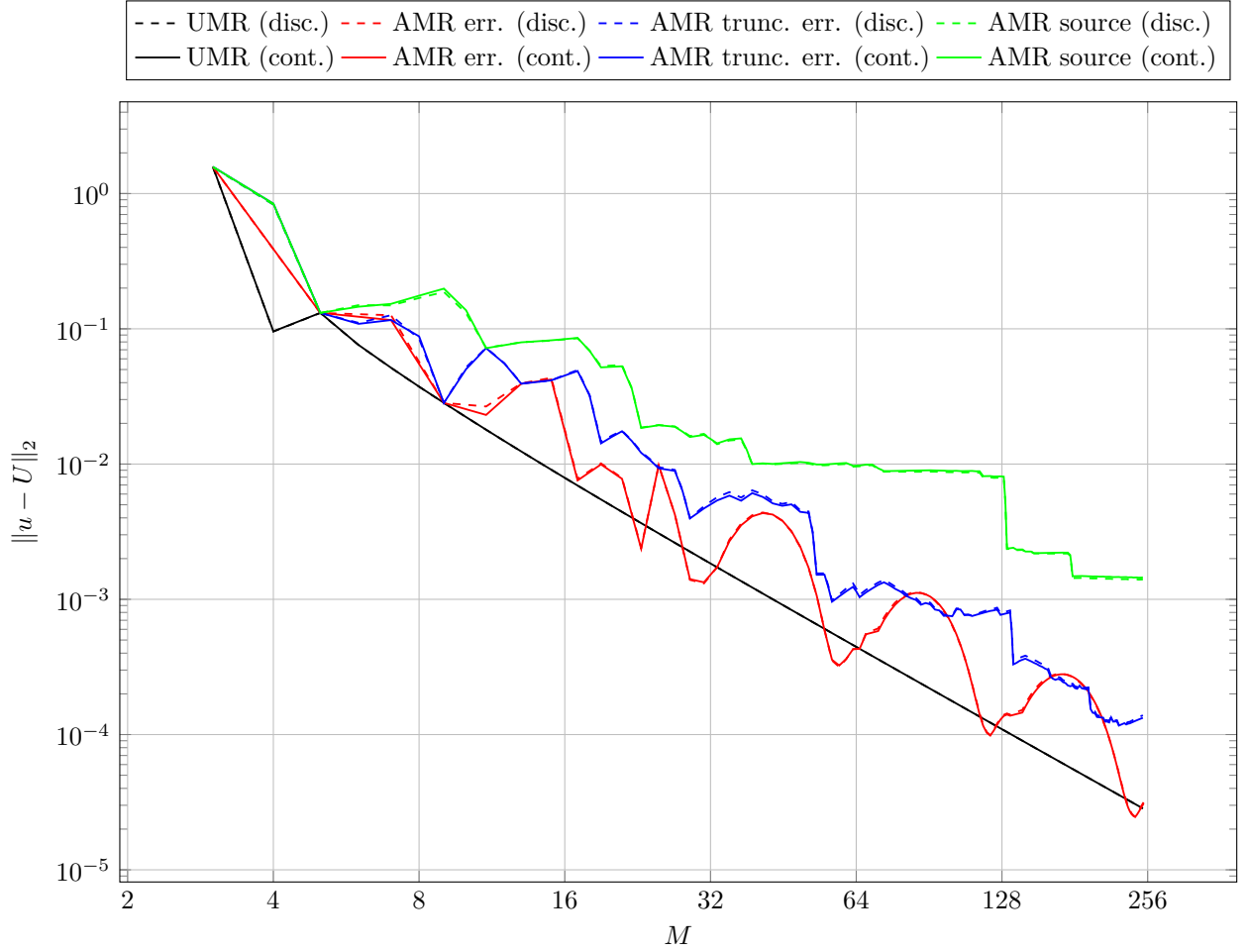


Figure 4: Comparison between the convergence of the numerical solution $U(x)$ with uniform mesh refinement (UMR) and adaptive mesh refinement (AMR) on the problem $u_{xx} = f(x)$ on $x \in [0, 1]$ with analytical solution $u(x) = \exp(-(x - 1/2)^2/0.1)$. The adaptive refinement is done using three different 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 the convergence of the two methods using mesh refinement as we did in section 1.

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 approximate the spatial derivative with central finite differences. Introducing $v_m(t), m = 0, \dots, M$ as approximations to $u(x_m, t)$, the result is now that we have gone from a PDE to a system 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, using e.g. Euler's method. For convenience we employ the θ -method,

General θ method,

and obtain

$$\text{method for heat equation.} \tag{5}$$

Integrating with constant time step k results in a uniform grid for the time interval as well. As for the spatial grid, we divide the time interval into N equidistant points, so that we approximate the solution at N finite times $t_n = nk, \quad n = 1, \dots, N, \quad k = 1/(N - 1)$.

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.

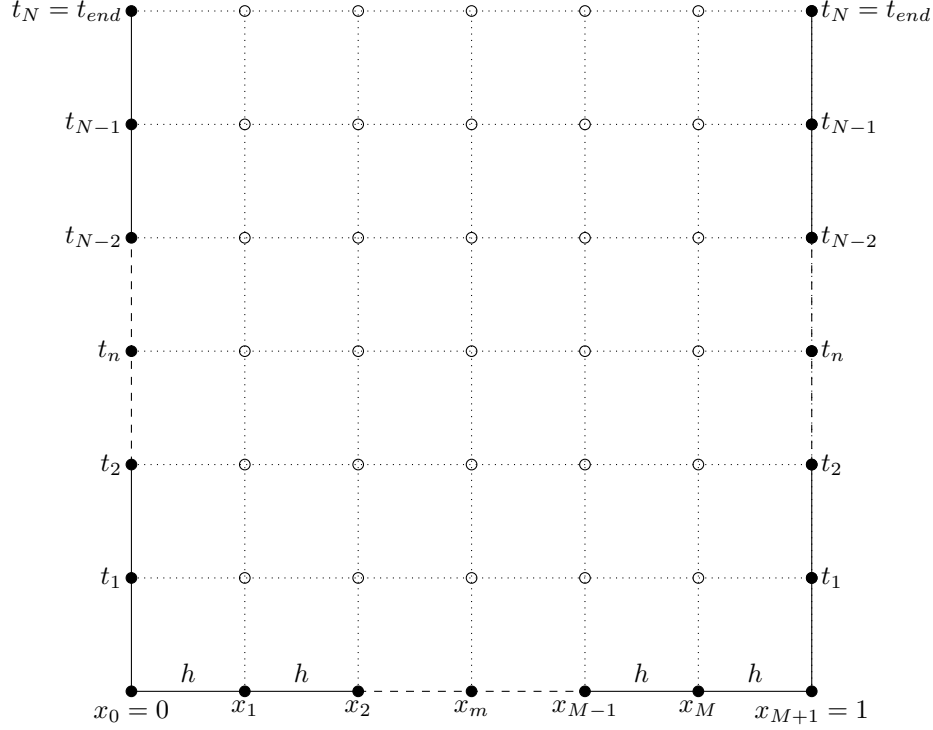
Imposing boundary conditions and some matrix equations here I guess.

We start by considering the heat equation with the following Neumann boundary conditions and initial condition,

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

2.2 Convergence and mesh refinement

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 now consider the heat equation with a set of boundary and initial conditions for which the analytical solution is known. Specifically we consider

$$u_t = u_{xx}, \quad u(0, t) = u(1, t) = 0, \quad u(x, 0) = \sin(\pi x), \quad (7)$$

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

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

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 $t = t_{end}$. The number of time steps N is kept fixed, and compute both the L_2 discrete relative error. For equation (7) we also compute the l_2 continuous relative error, and 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 .

3 Inviscid Burgers' equation

In this section we briefly inspect and solve the inviscid Burgers' equation, and evaluate the breaking which this equation exhibits.

Notes: -Also semi-discretization, but integrate using off-shelf routine -Breaking/shock formation: Numerical vs theoretical

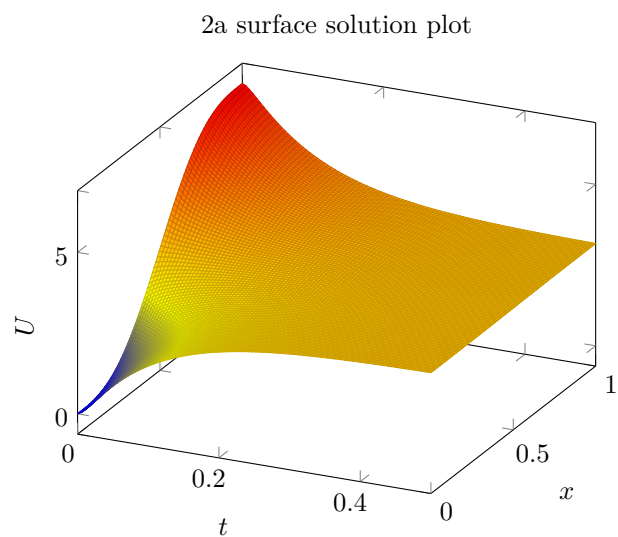
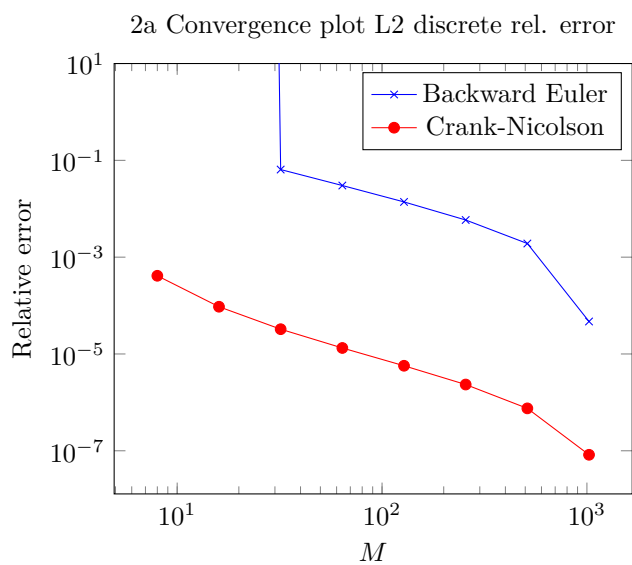
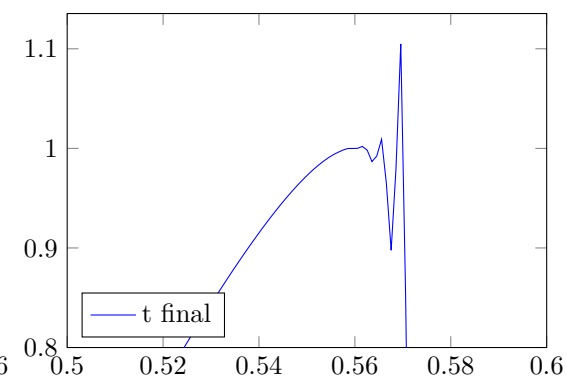
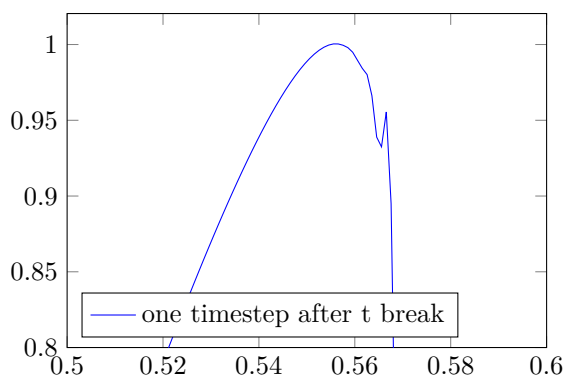
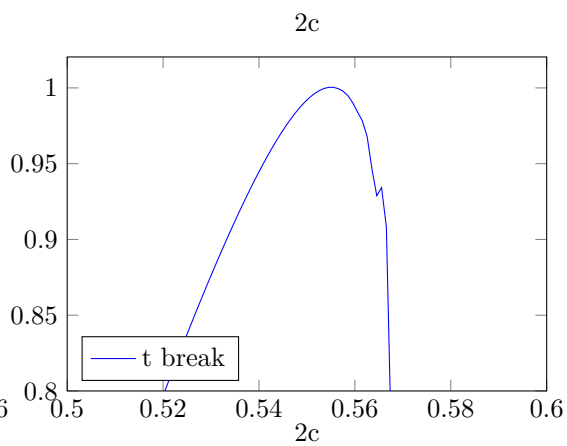
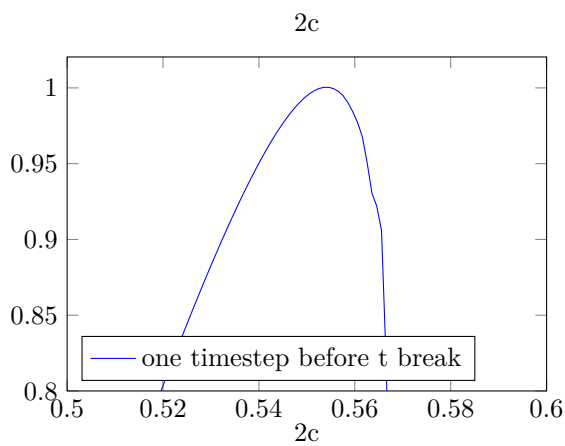
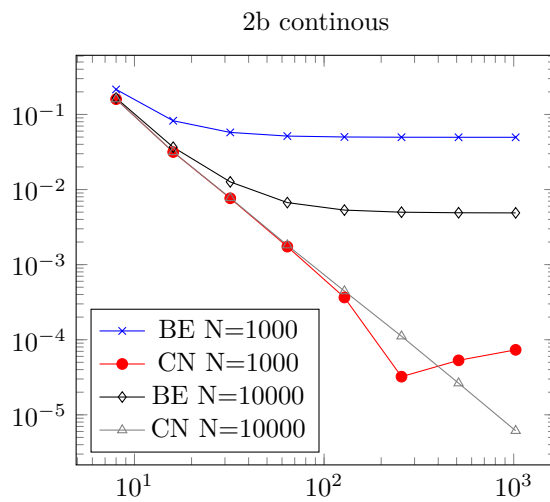
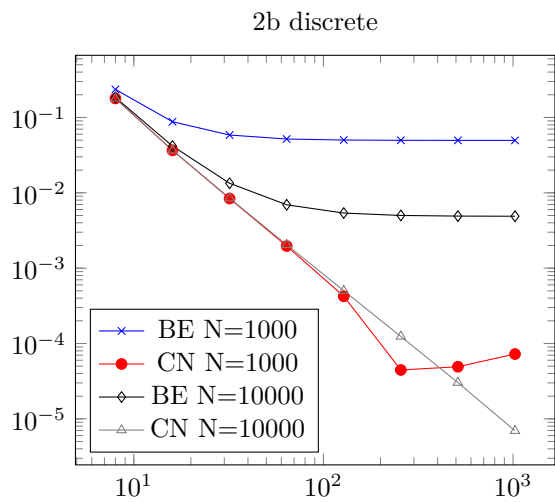


Figure 5: I am a surface plot, Hooray :)





4 Laplace equation in two dimensions

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

$$u_{xx} + u_{yy} = 0, (x, y) \in \Omega := [0, 1]^2, \quad (9)$$

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 9 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 9.

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 ??,

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

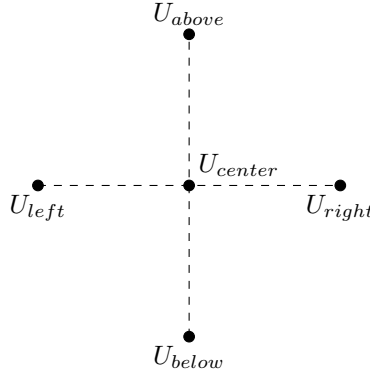


Figure 6: The five-point stencil corresponding to central difference differentiation in both the x - and y -direction.

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.

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

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 ??.

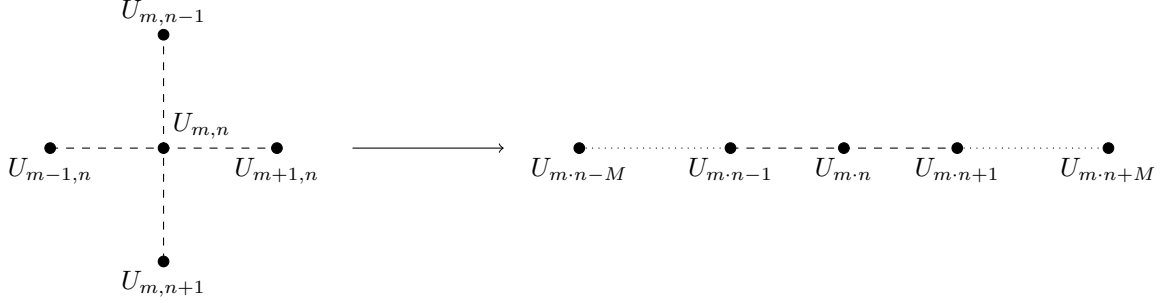


Figure 7: By flattening the five-point stencil, we can write the system of equations, which is now on the form $AU = 0$.

We thus write

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

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

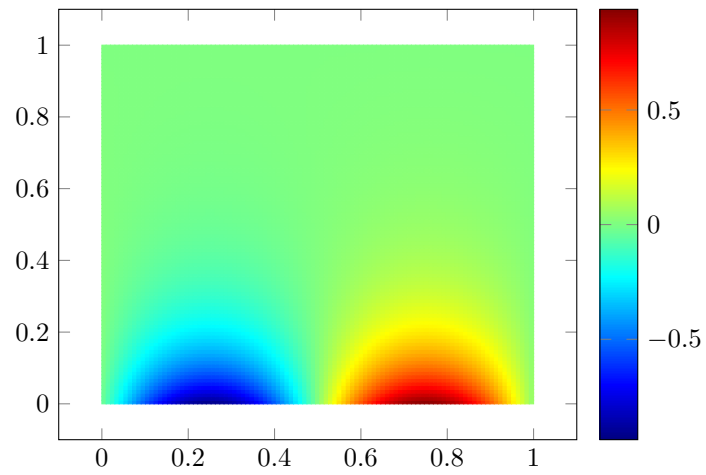


Figure 8: The numerically computed solution to the Laplace equation.

5 Linearized Korteweg-De Vries equation in one dimension

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

$$\frac{\partial u}{\partial t} + \left(1 + \pi^2\right) \frac{\partial u}{\partial x} + \frac{\partial^3 u}{\partial x^3} = 0 \quad (t \geq 0) \quad (-L/2 \leq x \leq +L/2), \quad (10)$$

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

$$u(x, t) = \sum_{n=-\infty}^{+\infty} c_n(t) \exp(ik_n x) \quad (11)$$

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 (10) 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

$$c_n(t) = c_n(0) \exp \left(-i \left(\left(1 + \pi^2\right) k_n - k_n^3 \right) t \right). \quad (12)$$

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 (10), 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 theta method. This gives the final system of difference equations for the numerical solution

$$\frac{U_m^{n+1} - U_m^n}{\Delta t} = (1 - \theta)F(U^n) + \theta F(U^{n+1}), \quad (13)$$

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

$$(I - \theta \Delta t A) U^{n+1} = (I - (1 - \theta) \Delta t A) U^n, \quad (14)$$

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 (14) repeatedly to step forward in time. Note that with the constant time step Δt , all matrices in equation (14) 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 (11). 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

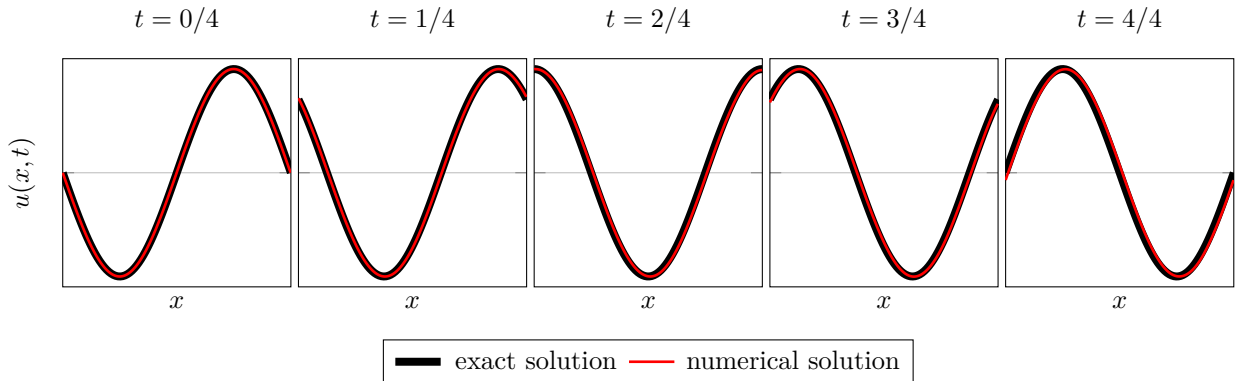


Figure 9: Comparison between the time evolution of the exact solution $u(x, t) = \sin(\pi(x - t))$ and the numerical solution from the Crank-Nicholson method with $\Delta x = 1/799$ and $\Delta t = 1/99$.

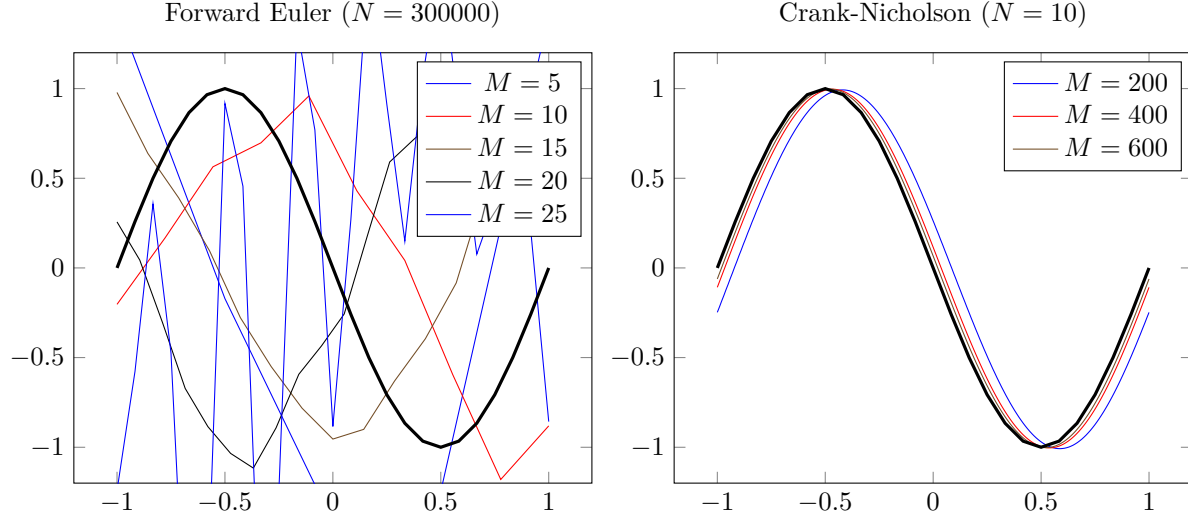


Figure 10

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

$$U_m^n = U(x_m, t_n) = \sum_l C_l^n \exp(ik_l x_m). \quad (15)$$

Consider now a single Fourier mode $C_l^n \exp(ik_l x_m)$ in this series. Inserting it into equation (13), 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

$$G_l = \frac{1 + i\Delta t f(x)/2}{1 - i\Delta t f(x)/2} \implies |G_l| = 1. \quad (16)$$

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

$$G_l = 1 + i\Delta t f(x) \implies |G_l| = \sqrt{1 + \Delta t^2 f(k_l)^2}. \quad (17)$$

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

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

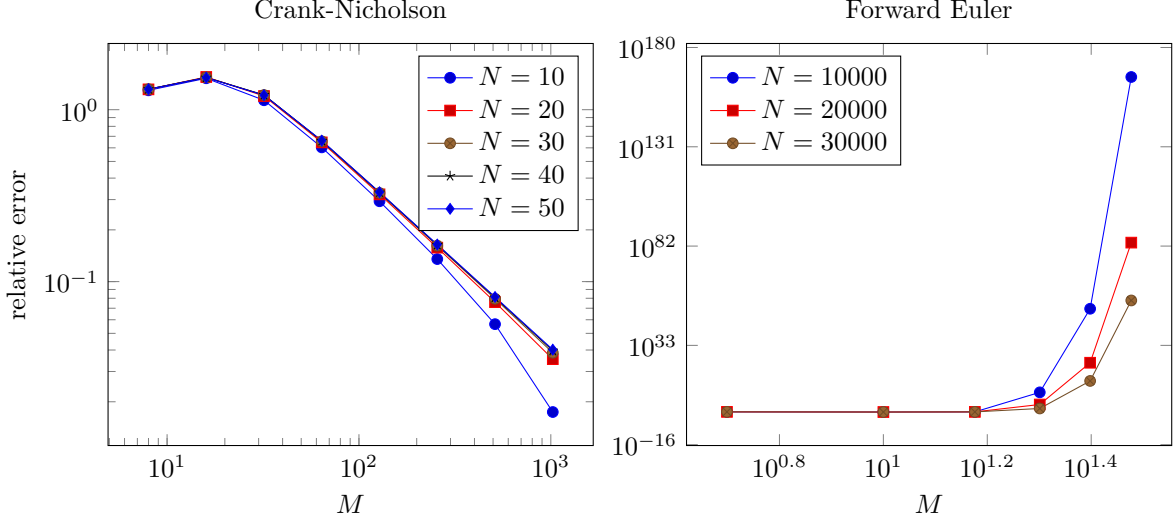


Figure 11: Convergence plot at $t = 1$, comparison between Crank-Nicholson and Forward Euler

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 greater 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 (11) 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 (12) 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 (10) that gives the exponential in equation (12) 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 (17). 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 (16).

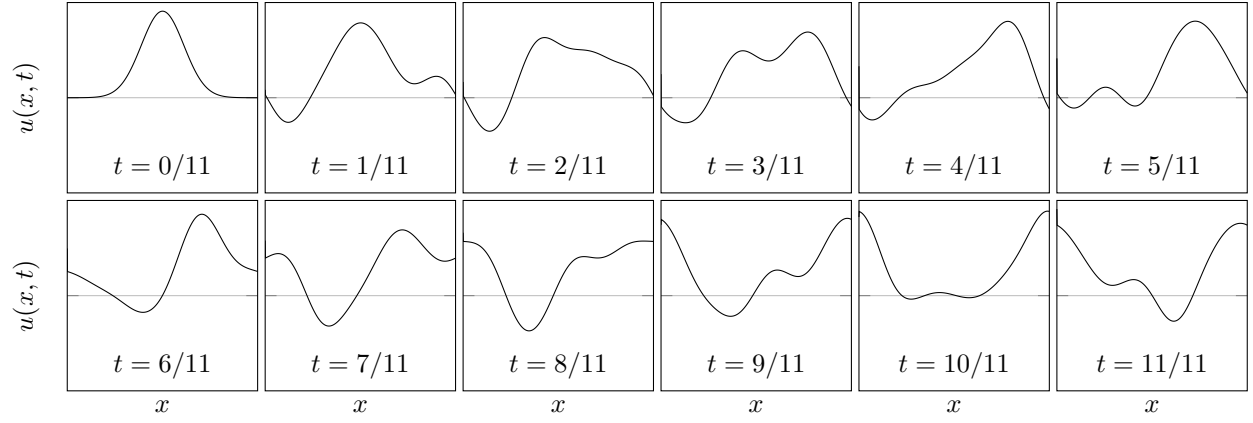


Figure 12: Time evolution of a initial gaussian $u(x, 0) = \exp(-x^2/0.1)$ computed from the Crank-Nicholson method on a grid with $M = 800$ points in space and $N = 100$ points in time.

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.

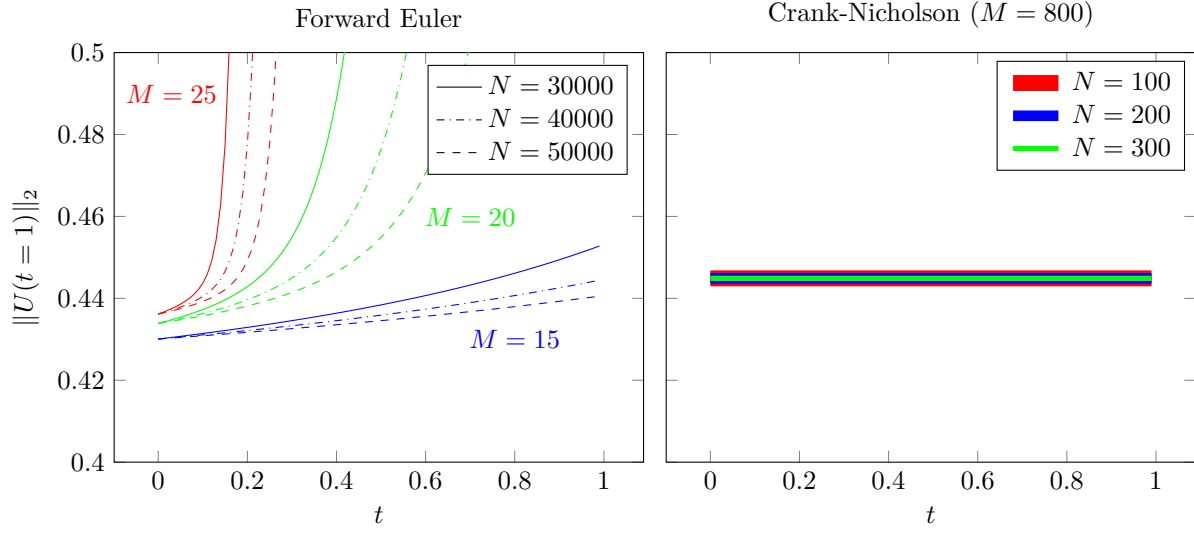


Figure 13: Time evolution of the discrete L_2 -norm of the initial gaussian $u(x, 0) = \exp(-x^2/0.1)$ computed from the Crank-Nicholson method and the Euler-method on different grids.

6 Poisson equation in one dimension finite element method

In this section, we will again solve the Poisson equation

$$-\frac{\partial^2 u}{\partial x^2} = f(x), \quad u(a) = \alpha, \quad u(b) = \beta, \quad (a \leq x \leq b) \quad (18)$$

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

6.1 Analytical solution

Same as in equation (2), but with $f(x) \rightarrow -f(x)$.

$$u(x) = C_1 + C_2 x - \int^x dx' \int^{x'} dx'' f(x''), \quad (19)$$

6.2 Numerical solution using finite element method

Write

$$u(x) = \hat{u}(x) + r(x), \quad \text{with} \quad \hat{u}(a) = \hat{u}(b) = 0 \quad \text{and} \quad r(x) = \alpha \frac{x-b}{a-b} + \beta \frac{x-a}{b-a} \quad (20)$$

Now multiply equation (18) by a function $v(x)$, integrate both sides from a to b and use integration by parts on the left and $\hat{u}(a) = \hat{u}(b) = 0$ to drop the boundary term. This gives the **weak formulation** of the problem: find \hat{u} such that for all v

$$\int_a^b dx \hat{u}'(x) v'(x) = \int_a^b dx f(x) v(x) - \int_a^b dx r'(x) v'(x) \quad (21)$$

Equation (21) is fully equivalent to equation (18).

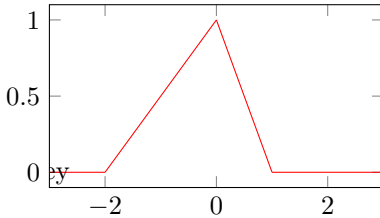
The approximation lies in seeking a solution $U(x) = \hat{U}(x) + R(x) \approx u(x)$ belonging to the space of piecewise linear functions. Divide the interval $[a, b]$ into the finite elements

$$a = x_0 < x_1 < \dots < x_M < x_{M+1} = b.$$

and let $U(x)$ be piecewise linear between these points. This leads to the weak formulation

$$\int_a^b dx \hat{U}'(x) V'(x) = \int_a^b dx f(x) V(x) - \int_a^b dx R'(x) V'(x) \quad (22)$$

Expand $U(x) = \sum_{i=0}^{M+1} U_i \varphi_i(x)$ and $V(x) = \sum_{i=0}^{M+1} V_i \varphi_i(x)$ in

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


and insert back to get

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

or $V^T A \hat{U} = V^T F$ where then

$$A \hat{U} = F \quad \text{with} \quad A_{ij} = \int_a^b dx \varphi'_i(x) \varphi'_j(x) \quad \text{and} \quad F_j = \int_a^b dx \varphi_j(x) \quad (23)$$

The A integrals are straightforward to differentiate 6.2:

$$\begin{aligned} A_{00} &= \frac{1}{x_1 - x_0} \\ A_{ii} &= \frac{1}{x_i - x_{i-1}} + \frac{1}{x_{i+1} - x_i} \\ A_{M+1 M+1} &= \frac{1}{x_{M+1} - x_M} \\ A_{ii+1} &= A_{i+1i} = \frac{-1}{x_{i+1} - x_i} \end{aligned} \quad (24)$$

We now impose $\hat{U}(a) = \hat{U}(b) = 0$ by removing the first and last entries in the matrix equation *after* calculating the right side. This gives a $M \times M$ equation. Then we reconstruct the full solution by adding α and β as the first and last entries to get $M + 2$ points. From U , we reconstruct $U(x) = \sum_{i=0}^{M+1} U_i \varphi_i(x)$.

6.2.1 Uniform refinement

We now test our finite element method on four problems in figure 13 with uniformly sized elements. Except on the first problem, errors are distributed non-evenly across the elements. By using uniform elements, we therefore waste a lot of space by using many elements in regions where the solution is already well known.

6.2.2 Adaptive refinement

This is the motivation for doing adaptive refinement of the elements. This time, we will do it a little differently than in ONE. Instead of splitting only one element between each iteration of the numerical solution, we now present two strategies that splits a dynamic number of elements so that all errors reduce below some given level in every iteration.

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 prefactor ensures that intervals are split also when all errors are equal up to machine precision, so the refinement does not halt.

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 problems using the average error strategy. Finally, in figure 14, we compare the convergence of uniform refinement and the two adaptive refinement strategies. Note that the adaptive refinement always yields a better result, unlike in ONE!

Related to the fact that the basis functions “cover the whole domain”?

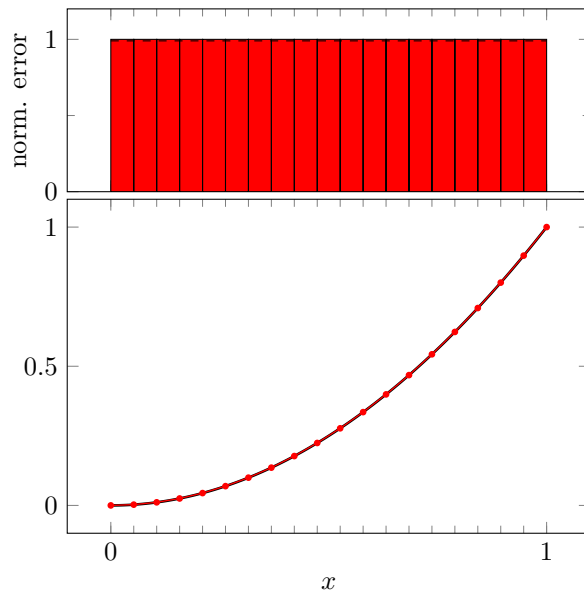
Comment on uniform = adaptive in 1st problem?

Comment on which numerical integration procedure to use?

Comment on how to handle diverging f in 4th problem?

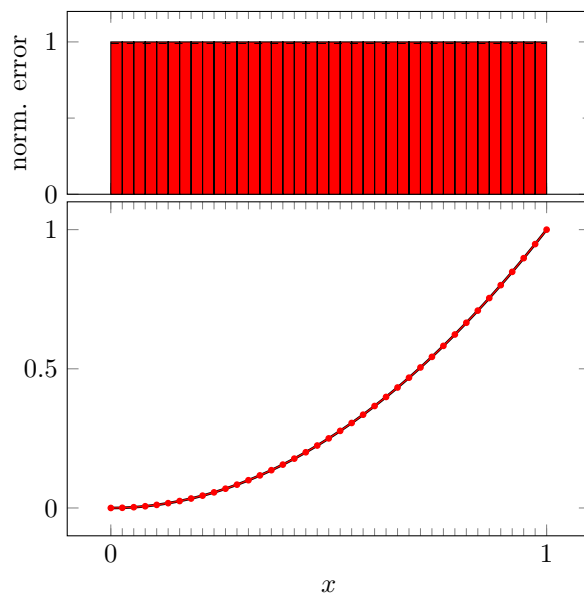
$$f(x) = -2$$

$$M = 20$$



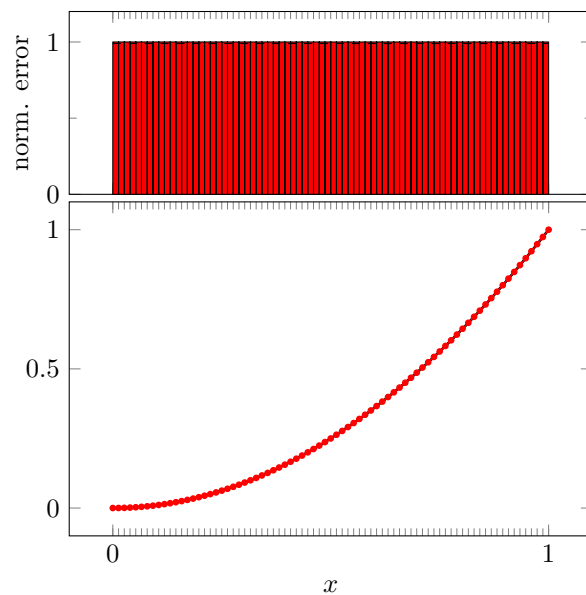
$$f(x) = (40000x^2 - 200) \exp(-100x^2)$$

$$M = 39$$



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

$$M = 77$$



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

$$M = 153$$

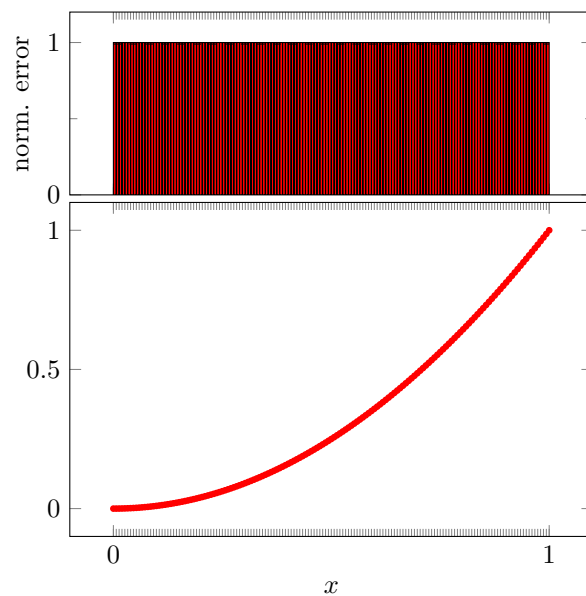
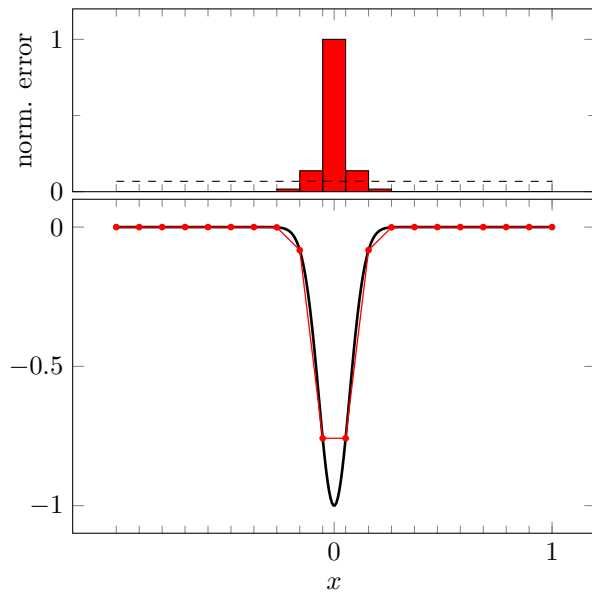


Figure 13: Uniform refinement

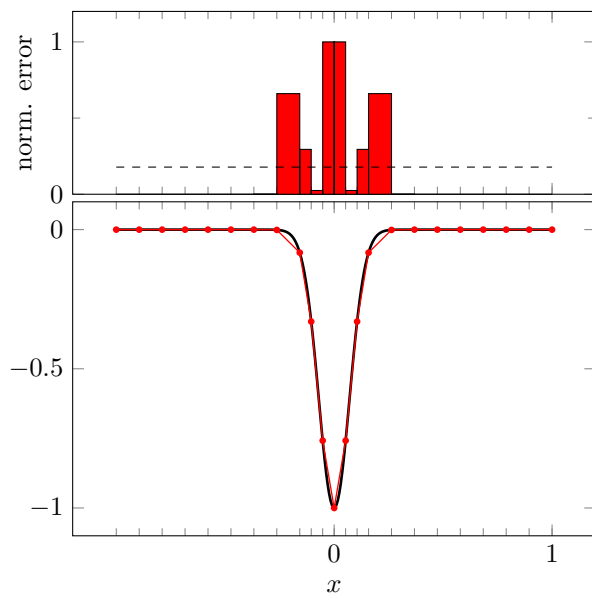
$$f(x) = -2$$

$$M = 20$$



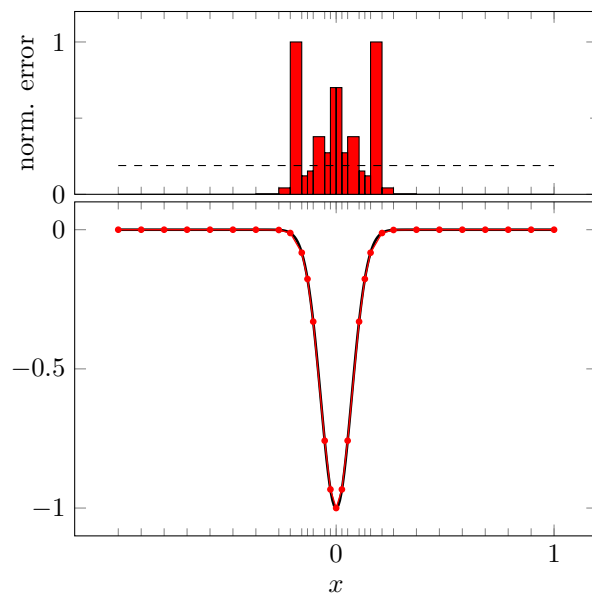
$$f(x) = (40000x^2 - 200) \exp(-100x^2)$$

$$M = 23$$



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

$$M = 29$$



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

$$M = 37$$

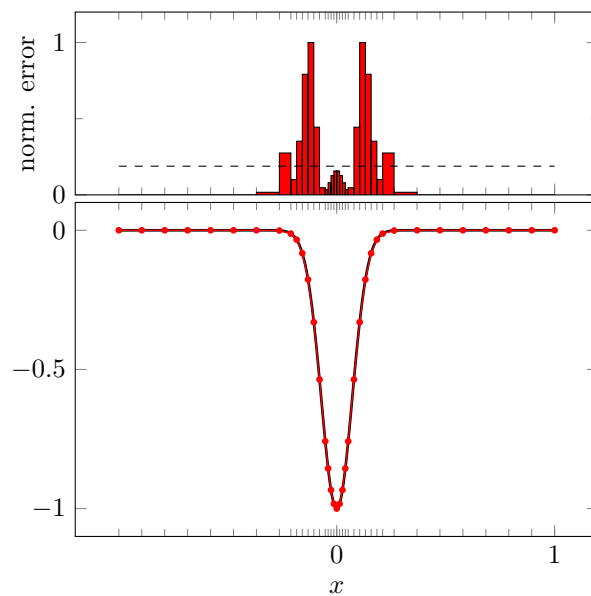


Figure 13: Adaptive refinement, average strategy

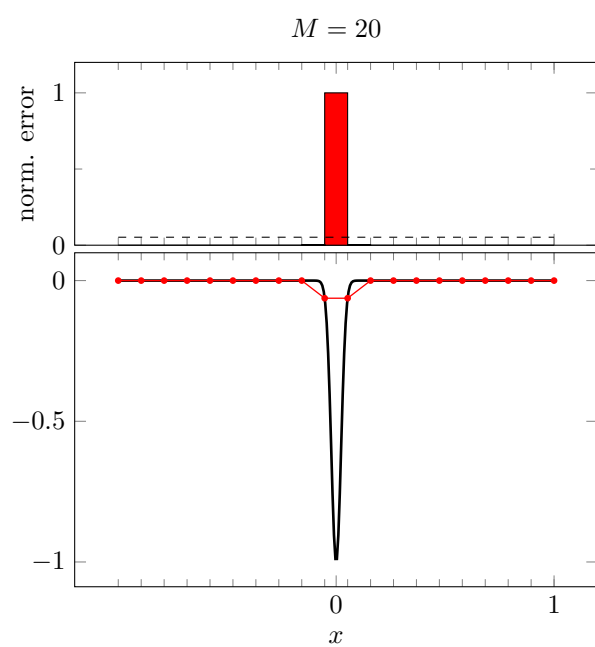


Figure 14: Convergence plot, comparison of three strategies

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>]