Semester Project in TMA4212

Thorvald M. Ballestad
Jonas Bueie
Knut Andre Grytting Prestsveen
Herman Sletmoen

April 30, 2021

Contents

1 Poisson equation in one dimension

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

$$\frac{\partial^2 u}{\partial x^2} = f(x) \qquad (0 < x < 1) \tag{1}$$

subject to a source term $f(x)$ and different combinations of Dirichlet and Neumann 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

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

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 homogenuous equation $u_{xx} = 0$ and a solution to the inhomogenuous equation $u_{xx} = f(x)$.

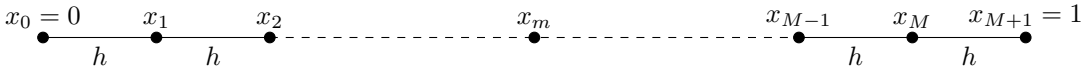
Note that if *two* Neumann boundary conditions $u_x(0) = a$ and $u_x(1) = b$ are imposed, then the solution $u(x)$ is unique only up to a constant. If $u(x)$ is a solution to the boundary value problem defined by equation (1) with $u_x(0) = a$ and $u_x(1) = b$, then also $(u + C)_{xx} = u_{xx}$ in the 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

In order to achieve a numerical solution to the problem, we first consider the boundary value problem defined by equation (1), subject to the boundary conditions

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

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



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

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

Discarding terms of $\mathcal{O}(h^2)$ and denoting the approximation of the solution u_m as U_m, we get the finite difference formula

$$\frac{U_{m-1} - 2U_m + U_{m+1}}{h^2} = f_m \qquad (1 \leq m \leq M - 1). \tag{3}$$

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

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

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

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

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

$$\begin{bmatrix} 1 & & & & & \\ +1/h^2 & -2/h^2 & +1/h^2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & +1/h^2 & -2/h^2 & +1/h^2 & \\ & & +1/2h & -2/h & +3/2h & \end{bmatrix} \begin{bmatrix} U_0 \\ U_1 \\ \vdots \\ U_M \\ U_{M+1} \end{bmatrix} = \begin{bmatrix} a \\ f(x_1) \\ \vdots \\ f(x_M) \\ b \end{bmatrix}, \tag{4}$$

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 instead find the least-squares solution to the matrix equation. [5]

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

$$u(x) = C_1 + C_2 x + \frac{1}{3!}x^3 - \frac{1}{4\pi^2}\cos(2\pi x)).$$

In figure 1, we present numerical solutions for three different combinations of boundary conditions. To the right of each solution plot is the corresponding convergence plot, resulting from refinement of the uniform grid. The convergence plots show the computed relative errors at varying grid resolution, together with the expected convergence rate.

To find the expected convergence rate, we start by insert the exact solution into the finite difference formula (3). This gives rise to an additional term, the *local truncation error* τ_m, since the equality in the formula does not hold for the exact solution. We get that

$$\tau_m = \frac{u_{m-1} - 2u_m + u_{m+1}}{h^2} - f_m. \tag{5}$$

We now Taylor expand the terms around u_m, and the only difference between the expansion of $u_{m-1} = u(x_m - h)$ and the expansion of $u_{m+1} = u(x_m + h)$, is the sign in front of the odd-order terms, which

therefore cancel. The expansion of the truncation error is thus

$$\begin{aligned}\tau_m &= \frac{1}{h^2}\left(-2u_m + 2u_m + 2\frac{h^2}{2}\partial_x^2 u_m + 2\frac{h^4}{24}\partial_x^4 u_m + \mathcal{O}(h^6)\right) - f_m \\ &= \partial_x^2 u_m + \frac{h^2}{12}\partial_x^4 u_m + \mathcal{O}(h^4) - f_m \\ &= \frac{h^2}{12}\partial_x^4 u_m + \mathcal{O}(h^4) \\ &= \mathcal{O}(h^2).\end{aligned}$$

For relating the local error τ_m to the global error and the expected convergence rate, we will for brievety just provide a slightly rough discussion. By rearanging (5) and subtracting it from equation (3), we can express the discretization error $e_m = U_m - u_m$ implicitly by

$$\frac{1}{h^2}\delta^2(u_m - U_m) = \frac{1}{h^2}\delta^2 e_m = \tau_m.$$

Defining the vectors $E = [e_1, \dots, e_m]$ and $\tau = [\tau_1, \dots, \tau_m]$, we can write this on matrix form so that

$$A'E = \tau.$$

The matrix A' can be shown to be invertible, and for the matrix norms subordinate to both the L_2 continous function norm, and the l_2 discrete vector norm, one can also show that $\|(A')^{-1}\|$ is bounded. This means that

$$\|E\| = \|(A')^{-1}\tau_m\| \leq \|(A')^{-1}\| \cdot \|\tau_m\| = C\|\tau\|.$$

Thus we can expect that the global error converges as the local truncation error.

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 $f(x_1) \to f(x_1) - a/h^2$. This approach is done in [6], for example, and 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 fictuous 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 $a \to a + hf(x_0)/2$. 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. This approach is also done in [6].

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 $x_i - x_{i-1} \neq$ 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.

To derive a nonuniform stencil for the second derivative at x_m, we proceed similarly to the uniform stencil. First approximate one derivative by stepping halfway left and right, landing at $x_{m-1/2}$ and $x_{m+1/2}$. Then

we approximate another derivative by stepping halfway to the sides again, landing at x_{m-1}, x_m and x_{m+1}. This yields

$$u''_m \approx \frac{u'_{m+1/2} - u'_{m-1/2}}{x_{m+1/2} - x_{m-1/2}} \approx \frac{2}{x_{m+1} - x_{m-1}} \left(\frac{u_{m+1} - u_m}{x_{m+1} - x_m} - \frac{u_m - u_{m-1}}{x_m - x_{m-1}} \right).$$

Assuming Dirichlet boundary conditions, the nonzero entries of A (indexed from zero) becomes

$$A_{0,0} = A_{M+1,M+1} = 1 \qquad A_{m,m-1} = \frac{2}{x_{m+1} - x_{m-1}} \frac{1}{x_m - x_{m-1}}$$

$$A_{m,m} = \frac{-2}{x_{m+1} - x_{m-1}} \left(\frac{1}{x_m - x_{m-1}} + \frac{1}{x_{m+1} - x_m} \right) \qquad A_{m,m+1} = \frac{2}{x_{m+1} - x_{m-1}} \frac{1}{x_{m+1} - x_m}.$$

The job is then once again to solve the system $AU = b$. Note that the stencil reduces to the one in equation (4) when $x_m - x_{m-1} = x_{m+1} - x_m = h$, as it should.

To do adaptive mesh refinement, we will

1. Start with a coarse uniform grid, 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

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

   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.

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

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

   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 knowlege 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 $M \to M + 2$. 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 **??**, 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.

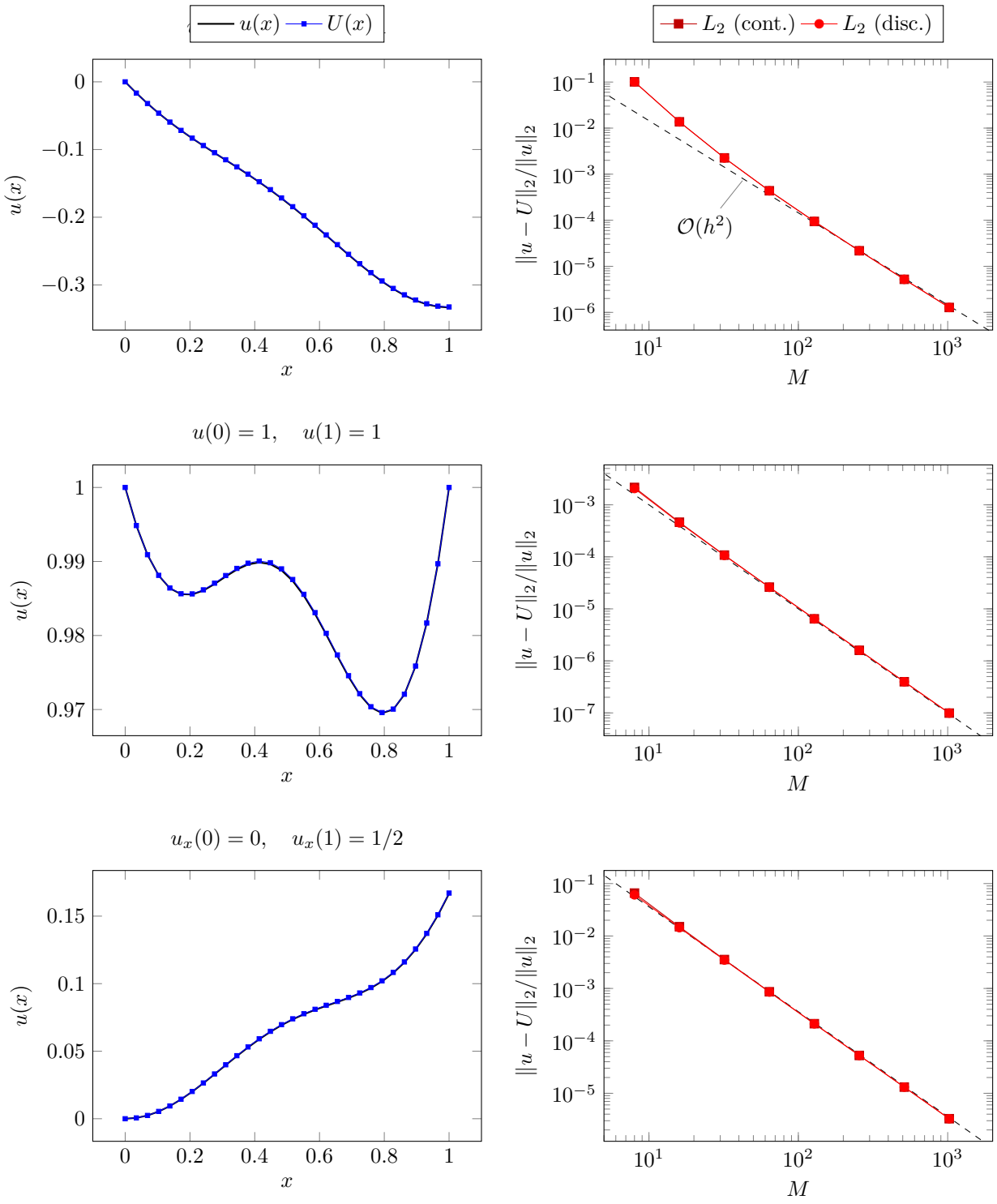


Figure 1: The left plots show analytical solutions $u(x)$ and numerical solutions $U(x)$ with $M = 30$ grid points for $u_{xx} = x + \cos 2\pi x$ subject to three different boundary conditions. The right plots show convergence plots corresponding to the same boundary conditions, where the error is measured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

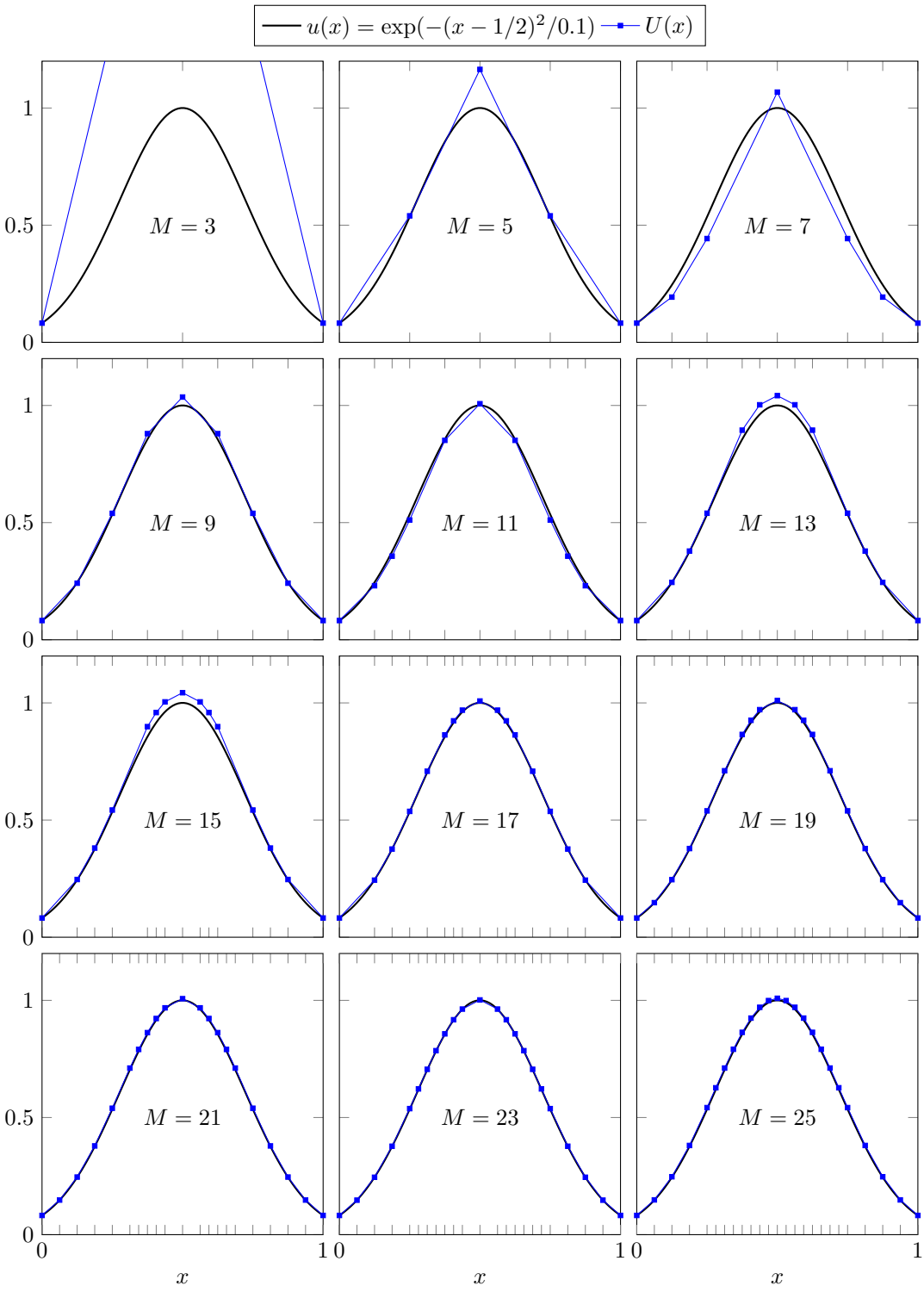


Figure 2: During adaptive mesh refinement (AMR) with the error strategy (strategy number 1), the interval with the largest error $\int \mathrm{d}x|u(x) - U(x)|$ is split in half. Here, $u(x) = \exp(-(x-1/2)^2/0.1)$ is the symmetric solution to whichever Poisson equation has $f(x) = u_{xx}$ on $x \in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

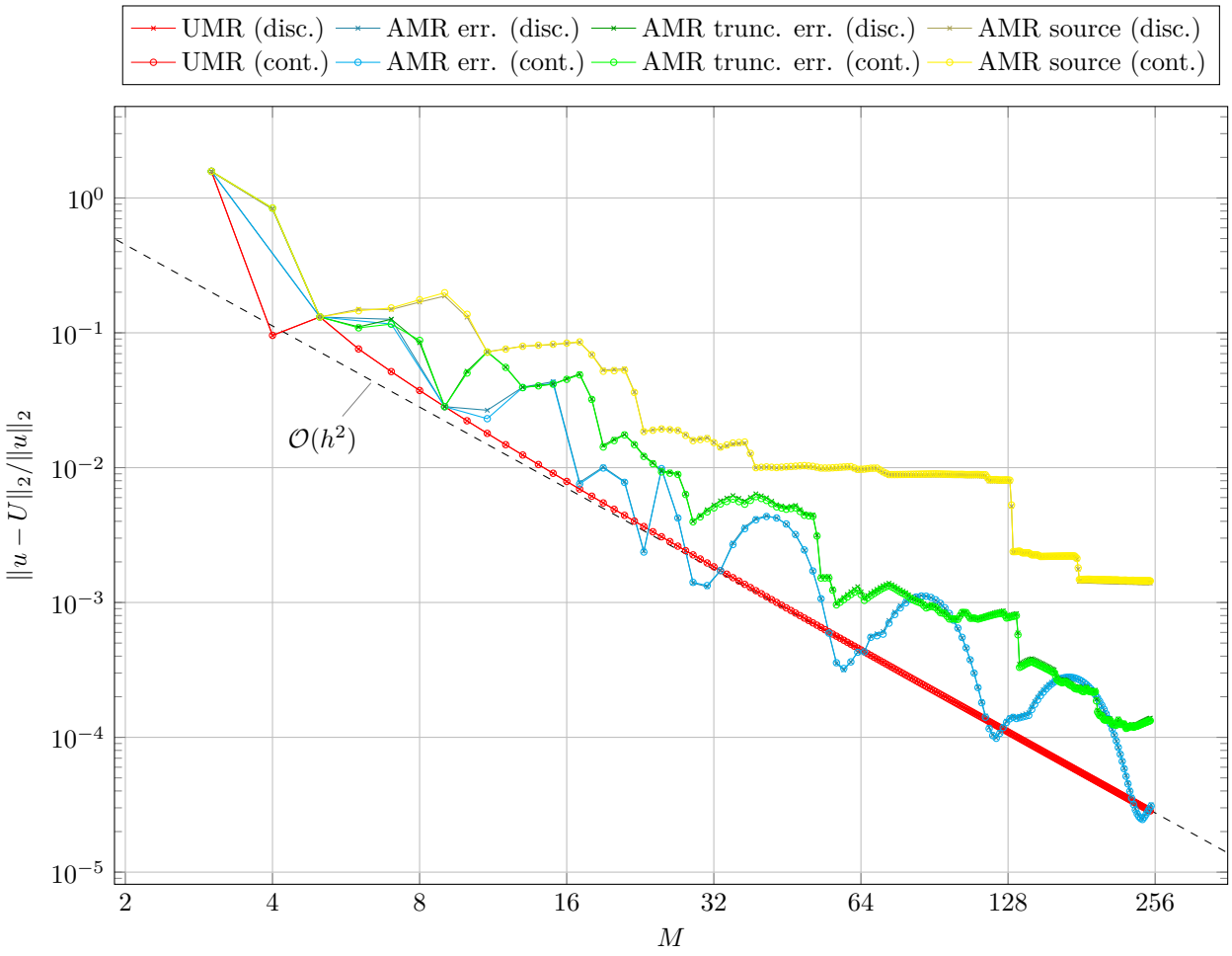


Figure 3: Comparison between the convergence of the numerical solution $U(x)$ with uniform mesh refinement (UMR) and adaptive mesh refinement (AMR) on the problem $u_{xx} = f(x)$ on $x \in [0, 1]$ with analytical solution $u(x) = \exp(-(x - 1/2)^2/0.1)$. The adaptive refinement is done using three different 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 $\int \mathrm{d}x|f(x)|$. Errors $\|u - U\|_2$ are measured with the continuous and discrete L_2-norm. Note that a cross inside a circle means that the discrete and continous error measurements agree.

2 Heat equation in one dimension

In this section, we consider the heat equation for $u = u(x,t)$ in one spatial dimension,

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

with either Neumann or Dirichlet boundary conditions. We solve it numerically using both the Backward Euler method and the Crank-Nicolson method, which as we will later see, are $\mathcal{O}(k+h^2)$ and $\mathcal{O}(k^2+h^2)$ methods respectively. Then we will analyze and compare their convergence using mesh refinement as we did in section 1. We will do refinement of the grids in both the x-direction and the t-direction, 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 continous. As in section 1, we divide the interval Ω into $M+2$ equidistant nodes with separation $h = 1/(M+1)$, so that we get a uniform grid with M internal nodes and two boundary nodes. We then express the spatial derivative using the central finite difference to get

$$u_t(x_m,t) = \frac{1}{h^2}\delta_x^2 u(x_m,t) + \mathcal{O}(h^2), \quad m = 0,...,M+1.$$

We now introduce the single variate functions $v_m(t)$ as the approximation to $u(x_m,t)$, at each node 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, using for instance one of the many standard schemes for ODEs such as Euler's method. For the sake of convenience we employ the θ-method, which for general ODEs $y' = g(y,t)$ is given as

$$\frac{y^{n+1}-y^n}{k} = \Big((1-\theta)g(y^n,t_n) + \theta g(y^{n+1},t_{n+1})\Big),$$

where k is the time step, and the value of θ determines the specific numerical scheme

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

We use a constant step size $k = 1/(N-1)$ in time, where N denotes the number of time steps, and the final uniform grid is illustrated in figure 4. This gives the approximate solution of $v_m(t)$ at $t_n = nk$, where $n = 0,\dots N-1$, and we denote the fully discretized approximation of $u(x_m,t_n)$ as U_m^n. For the heat equation this results in the following finite difference formula

$$\frac{U_m^{n+1}-U_m^n}{k} = (1-\theta)\frac{1}{h^2}\delta_x^2 U_m^n + \theta\frac{1}{h^2}\delta_x^2 U_m^{n+1}. \tag{6}$$

After organizing the terms, the θ-method for the 1D heat equation is then written compactely as

$$(1-\theta r\delta_x^2)U_m^{n+1} = \Big(1+(1-\theta)r\delta_x^2\Big)U_m^n, \tag{7}$$

where we have defined $r = k/h^2$.

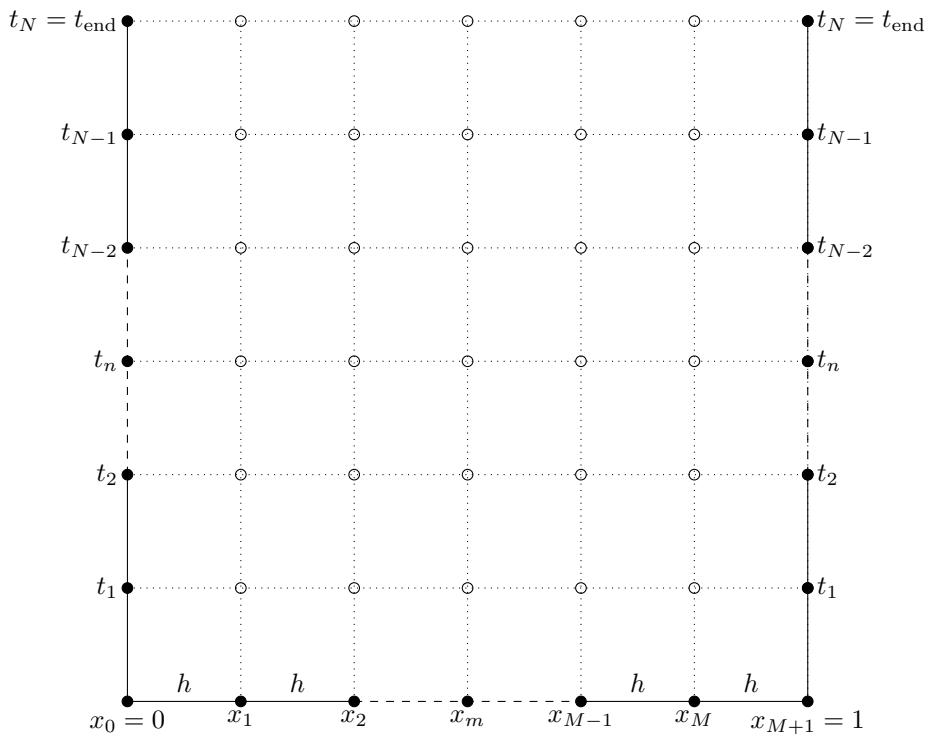


Figure 4: An illustration of how we divide the temporally and spatially continuous domain into a discrete, uniform grid. The filled circles correspond to values known through the initial and Dirichlet boundary conditions, and the empty circles to unknown values, to be determined using the finite difference methods. In the case of Neumann boundary conditions we will need to compute values at the boundaries as well.

To impose Dirichlet boundary conditions, $u(0,t) = \sigma$, $u(1,t) = \beta$, we substitute $U_0^{n+1} = \sigma$ and $U_{M+1}^{n+1} = \beta$ in equation (7) for $m = 1$ and $m = M$ to obtain

$$\begin{aligned}(1 + 2r\theta)\, U_1^{n+1} - r\theta U_2^{n+1} &= \big(1 - 2r(1-\theta)\big)\, U_1^n + r\,(1-\theta)\, U_2^n + r\sigma \quad (\text{for } m = 1),\\ (1 + 2r\theta)\, U_M^{n+1} - r\theta U_{M-1}^{n+1} &= \big(1 - 2r(1-\theta)\big)\, U_M^n + r\,(1-\theta)\, U_{M-1}^n + r\beta \quad (\text{for } m = M).\end{aligned}$$

We combine this with equation 7 for the remaining spatial nodes to write the system of equations in matrix form

$$(I - \theta r A)\, U^{n+1} = \big(I + (1-\theta) r A\big)\, U^n + \rho, \tag{8}$$

with

$$A = \begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix} \quad \text{and} \quad \rho = \begin{bmatrix} r\sigma \\ 0 \\ \vdots \\ 0 \\ r\beta \end{bmatrix}. \tag{9}$$

For Neumann boundary conditions, $u_x(0,t) = \sigma$, $u_x(1,t) = \beta$, we introduce fictitious nodes at $m = -1$ and $m = M + 2$, and approximate the first derivatives at the boundaries by

$$\frac{U_1 - U_{-1}}{2h} = \sigma \quad \text{and} \quad \frac{U_{M+2} - U_M}{2h} = \beta.$$

We then use these expressions to eliminate the fictitious nodes from equation (7) for $m = 0$ and $m = M + 1$ to get

$$(1 + 2r\theta)\, U_0^{n+1} - 2r\theta U_1^{n+1} = \big(1 - 2r(1-\theta)\big)\, U_0^n + 2r\,(1-\theta)\, U_1^n - 2hr\sigma \quad (\text{for } m = 0),$$
$$(1 + 2r\theta)\, U_{M+1}^{n+1} - 2r\theta U_M^{n+1} = \big(1 - 2r(1-\theta)\big)\, U_{M+1}^n + 2r\,(1-\theta)\, U_M^n + 2rh\beta \quad (\text{for } m = M + 1).$$

Now we can write the system of equations on the same matrix form (8), but with

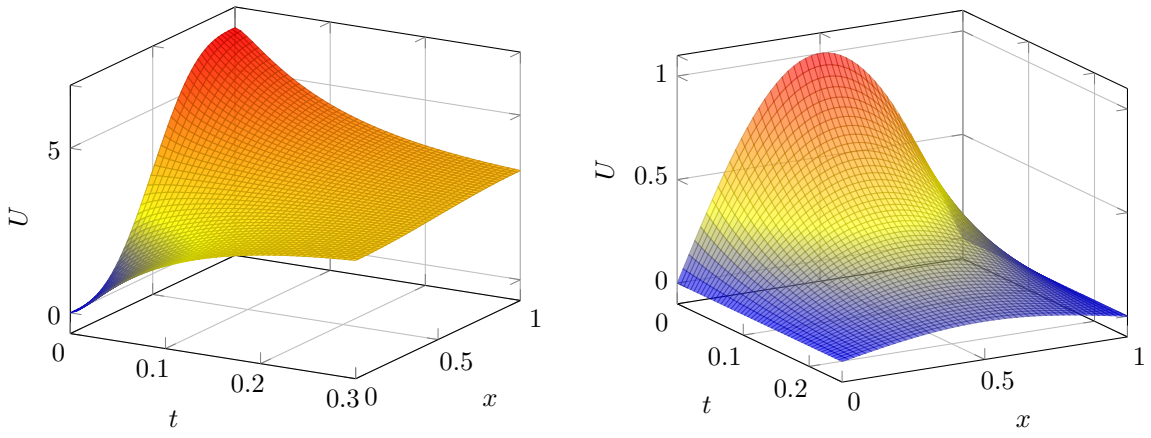
$$A = \begin{bmatrix} -2 & 2 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 2 & -2 \end{bmatrix} \quad \text{and} \quad \rho = \begin{bmatrix} -2rh\sigma \\ 0 \\ \vdots \\ 0 \\ 2rh\beta \end{bmatrix}. \tag{10}$$

Note that with Dirichlet conditions at both boundaries we only solve the equations for the internal spatial nodes $x_1 \dots x_M$ so that A in (9) is an $M \times M$ matrix. With Neumann conditions however we also need to solve for the boundary nodes, and A is in (10) an $(M + 2) \times (M + 2)$ matrix. In both cases though, all quantities on the right hand sides in (8) are known, i.e. the equations are on the form $A\vec{x} = \vec{b}$, and the known $\vec{b}$ is just written via a matrix-vector product for notational convenience. To solve the problem we now solve this system of equations at each time step, and since the matrices in both cases are tridiagonal, we represent them as sparse matrices and use a solver for sparse systems to save both memory and time.

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

$$u_x(0,t) = u_x(1,t) = 0, \quad u(x,0) = 2\pi x - \sin(2\pi x). \tag{11}$$

The computed solutions for $t \in [0, 0.3]$ is plotted in figure 5a, and qualitatively the solution behaves in accordance with what we 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.



(a) Neumann boundary conditions.

(b) Dirichlet boundary conditions.

Figure 5: Numerical solution of the one dimensional heat equation with different sets of boundary and initial conditions computed with Crank-Nicolson. Subfigure 5a shows the solution with the conditions specified in (11), and subfigure 5b is from solving with the conditions given in (12).

2.2 Convergence and mesh refinement

We will now demonstrate how the methods used in this section converge, depending on the mesh refinement strategy.

To find the expected convergence we start by finding the *local truncation error*. We denote the exact solution evaluated at the discrete grid points as $u_m^n = u(x_m, t_n)$. As mentioned in 1, inserting this into the finite difference formula (6) gives rise to an additional term τ_m^n, since the difference formula is not satisfied by the exact solution. The term τ_m^n is the *local truncation error*, and for our difference formula it is expressed as follows

$$\begin{aligned} k\tau_m^n &= u_m^{n+1} - u_m^n - (1-\theta)\frac{k}{h^2}\delta_x^2 u_m^n - \theta\frac{k}{h^2}\delta_x^2 u_m^{n+1} \\ &= (1 - \theta\frac{k}{h^2}\delta_x^2)u_m^{n+1} - u_m^n - \frac{k}{h^2}\delta_x^2 u_m^n + \theta\frac{k}{h^2}\delta_x^2 u_m^n \\ &= (1 - \theta\frac{k}{h^2}\delta_x^2)(u_m^{n+1} - u_m^n) - \frac{k}{h^2}\delta_x^2 u_m^n. \end{aligned}$$

Taylor expansion of all the terms around (x_m, t_n) gives

$$\begin{aligned} k\tau_m^n &= \left(1 - \theta k(\partial_x^2 + \frac{1}{12}h^2\partial_x^4 + \dots)\right)\left(k\partial_t + \frac{1}{2}k^2\partial_t^2 + \frac{1}{6}k^3\partial_t^3\right)u_m^n - k\left(\partial_x^2 + \frac{1}{12}h^2\partial_x^4 + \dots\right)u_m^n \\ &= \left(k\partial_x^2 + \frac{1}{2}k^2\partial_t^2 + \frac{1}{6}k^3\partial_t - \theta k^2\partial_t^2 - \frac{1}{2}k^3\partial_t^3 - k\partial_x^2 - \frac{1}{12}kh^2\partial_x^4 + \dots\right)u_m^n + \dots \\ &= \left(\frac{1}{2} - \theta\right)k^2\partial_t^2 - \frac{1}{12}kh^2\partial_x^4 u_m^n + \left(\frac{1}{6} - \frac{1}{2}\theta\right)k^3\partial_t^3 u_m^n + \dots \end{aligned}$$

Finally, dividing by k we find that

$$\begin{aligned} \tau_m^n &= \mathcal{O}(k + h^2) \quad \text{for } \theta \neq \frac{1}{2}, \\ \tau_m^n &= \mathcal{O}(k^2 + h^2) \quad \text{for } \theta = \frac{1}{2}. \end{aligned}$$

By repeating the same discussion as we did for the global discretization error in section 1, just extended to include also the temporal dimension, one can show the same result. Namely that the global error is expected to converge with the same rate as the local error τ_m^n, and so we end up with the convergence orders of $\mathcal{O}(k + h^2)$ for Backward Euler and $\mathcal{O}(k^2 + h^2)$, as advertised.

As in section 1 we now analyze the convergence of the numerical solution methods by doing mesh refinement. We restrict the analysis to just Backward Euler and Crank-Nicolson, which have different convergence order in the temporal direction. The reason for which we exclude the forward Euler method is that it is only stable under the condition that $r = k/h^2 < 1/2$, and such a condition would impose impractical restrictions on the grids and the refinement [6]. Backward Euler and Crank-Nicolson however, are unconditionally stable [6]. Later on when we study the Linearized Kortweg-de Vries equation in section **??**, we show this by applying Von Neumann stability analysis on the forward Euler and Crank-Nicolson methods.

We start by refining the spatial grid x_m sepparately, and we compute both the $l2$ discrete and the L_2 continous relative errors. For (11), however, the the analytical solution is not available in closed form, so we cannot compute the exact errors. In order to analyze convergence we therfore compute a reference solution, using a high resolution spatial grid with $M_{ref} = 10000$ spatial nodes, which we use in place of the analytical solution when computing the error. Since we are only refining in the spatial direction, we keep the number of time steps N fixed, and compute the numerical solution and the errors at the same point in time t with different values of M. This way the error in the time step will be constant throughout the refinement, allowing us to analyze the spatial convergence isolated.

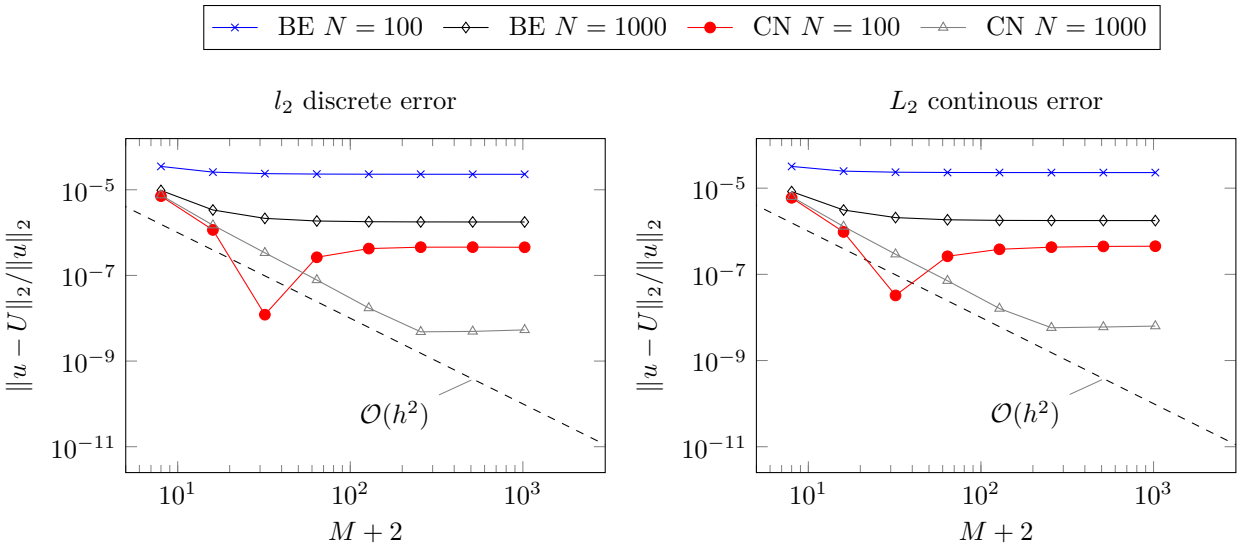


Figure 6: Convergence plots from uniformly refining the mesh in the spatial direction while keeping the number of time steps N constant. The equation solved is the one-dimensional heat equation $u_t = u_{xx}$ with the Neumann boundary conditions $u_x(0,t) = u_x(1,t) = 0$ and initial condition as listed in equation (11). The l_2 discrete and the L_2 continous relative errors is computed with respect to a numerical reference solution computed with $M_{ref} = 10000$, since the analytical solution for this problem is not available on closed form.

The resulting convergence rates from the refinement is plotted in figure 6, together with the expected convergence of $\mathcal{O}(h^2)$ in the spatial direction. We see that the errors for the most part fail to follow the expected curve, instead they quickly flatten and remain about constant throughout the refinement. Crank-Nicolson with $N = 1000$ is the exception, but even it's curve flattens towards the end of the refinement when M gets large enough. A flattend curve means that the error is dominated by the time step error, and further refining the spatial grid gives in that case only diminishing returns. We also see that the error curve of the solution computed with Crank-Nicolson flattens later than those of the solutions computed with Backward Euler for the same value of N. This reflects the fact that Crank-Nicolson is one order more precise in the time step k. Also, with a higher N we would be able to see the spatial $\mathcal{O}(h^2)$ convergence better.

In in order to analyze the convergence further we now switch to a set of boundary and initial conditions for which we can solve the heat equation analytically and compute the error properly. Specifically we consider

$$u(0,t) = u(1,t) = 0, \quad u(x,0) = \sin(\pi x), \tag{12}$$

on the same domain $x \in [0,1] := \Omega$ and $t > 0$. Note that we now have Dirichlet boundary conditions, and we have plotted the numerical solution in 5b. The analytical solution, which can be calculated using sepparation of variables [1] [2], is readily available as

$$u(x,t) = \sin(\pi x)e^{-\pi^2 t}. \tag{13}$$

We now do the same spatial refinement for equation (12), however now we compute the errors with respect to the analytical solution. The resulting convergence plots are shown in figure 7a, and here the characteristics we saw in 6 are seen more clearly. Again, Crank-Nicolson with $N = 1000$ performs the best, and displays here second order convergence througout the whole refinement. The others start out with second order

[1] We ommit the calculation since this is a very standard introductory problem for solving partial differential equations with sepparation of variables.

convergence, but flattens as the temporal error starts to dominate, and also here we see that the error in Crank-Nicolson flattens later than the error in Backward Euler for the same value of N.

Now we proceed to do refinement in the t-direction, while keeping the spatial step size constant. I.e. we do the same as in the spatial refinement, but instead we fix M and vary N. The resulting convergence plot is show in figure 7b, and here the difference in the temporal convergence of the two methods becomes very apparent. The chosen fixed values for M are large enough so that the spatial error does not dominate, and we avoid the error curves flattening as N gets large. The error curves also seem to follow the expected convergence order.

Having investigated the spatial and temporal convergence sepparately, we now look at the convergence when refining in the x- and t-directions simultaneously. We start with refinement where the time step and spatial step are varied at equal rates, by keeping $c = k/h$ constant. Since we are now refining in both time and space, we want to express the convergence in terms of the system's number of degrees of freedom N_{dof}, which we also will have on the x-axis in the convergence plot to plot the error up against. Defining $M^* = M + 2$ denoting the total number of spatial nodes, we can write $N_{\text{dof}} = M^*N$. Using that $M^* = 1/h + 1$ and $N = 1/k + 1$ as well as our refinement restriction $c = k/h$, we get that

$$\begin{aligned} N_{\text{dof}} = M^*N &= \left(\frac{1}{k} + 1\right)\left(\frac{1}{h} + 1\right) \\ &= \left(\frac{1}{h} + 1\right)\left(\frac{1}{ch} + 1\right) \\ &= \frac{1}{ch^2} + \frac{1}{ch} + \frac{1}{h} + 1 \\ &= \mathcal{O}(\frac{1}{h^2}). \end{aligned}$$

To relate this to the convergence rates we again use that $c = k/h$ to get

$$\begin{aligned} \text{Backward Euler:} \quad \mathcal{O}(k + h^2) = \mathcal{O}(ch + h^2) = \mathcal{O}(h) = \mathcal{O}(N_{\text{dof}}^{-1/2}) \\ \text{Crank-Nicolson:} \quad \mathcal{O}(k^2 + h^2) = \mathcal{O}(c^2h^2 + h^2) = \mathcal{O}(h^2) = \mathcal{O}(N_{\text{dof}}^{-1}). \end{aligned}$$

The resulting convergence plots with computed error curves, as well as the expected convergence rates, is shown in figure 8a.

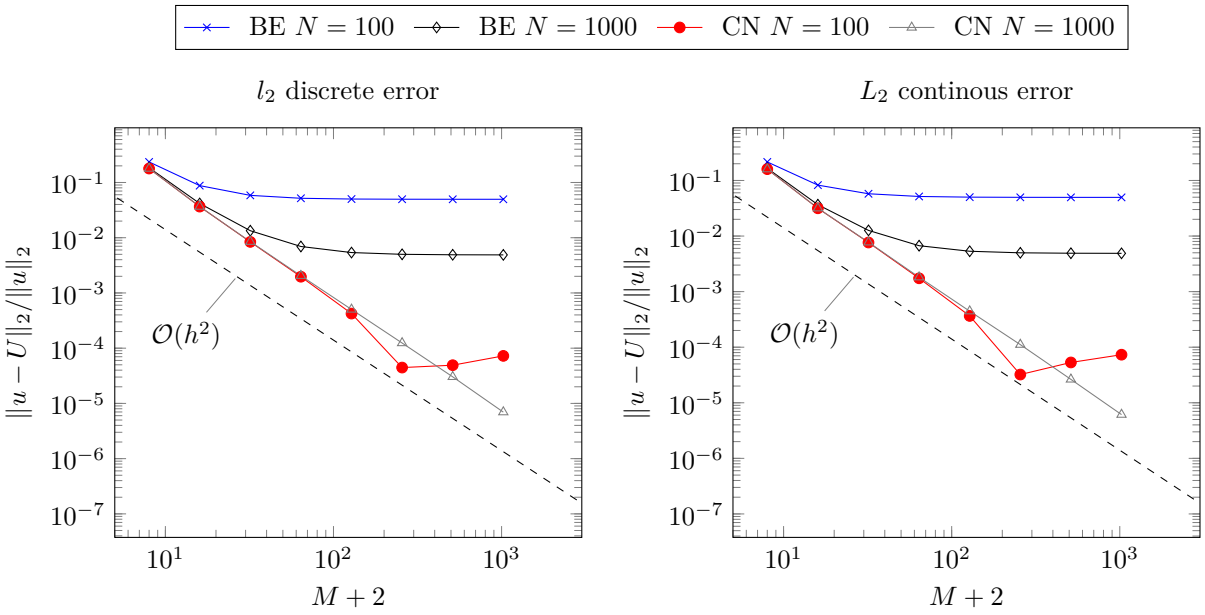
Finally we will do simultaneous refinement where we keep $r = k/h^2$ constant. For the degrees of freedom N_{dof} we now get

$$\begin{aligned} N_{\text{dof}} = M^*N &= \left(\frac{1}{k} + 1\right)\left(\frac{1}{h} + 1\right) \\ &= \left(\frac{1}{h} + 1\right)\left(\frac{1}{rh^2} + 1\right) \\ &= \frac{1}{rh^3} + \frac{1}{rh^2} + \frac{1}{h} + 1 \\ &= \mathcal{O}(\frac{1}{h^3}), \end{aligned}$$

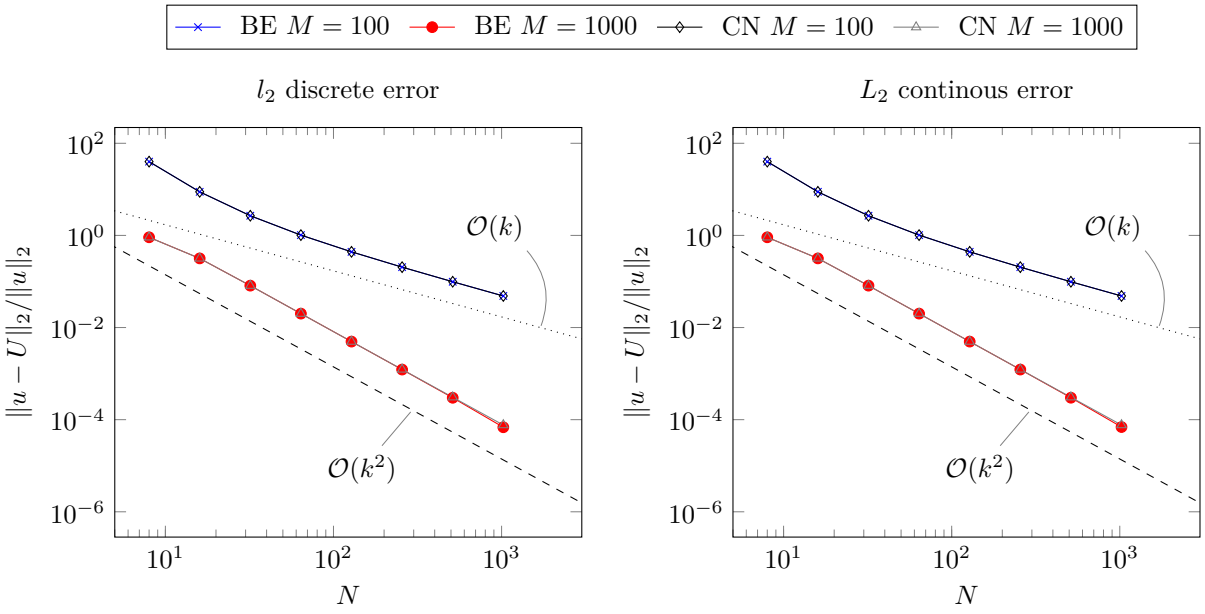
and for the convergence rates we get

$$\begin{aligned} \text{Backward Euler:} \quad \mathcal{O}(k + h^2) = \mathcal{O}(rh^2 + h^2) = \mathcal{O}(h^2) = \mathcal{O}(N_{\text{dof}}^{-2/3}) \\ \text{Crank-Nicolson:} \quad \mathcal{O}(k^2 + h^2) = \mathcal{O}(r^2h^4 + h^2) = \mathcal{O}(h^2) = \mathcal{O}(N_{\text{dof}}^{-2/3}). \end{aligned}$$

The resulting convergence plot for this final refinement is shown in figure 8b.

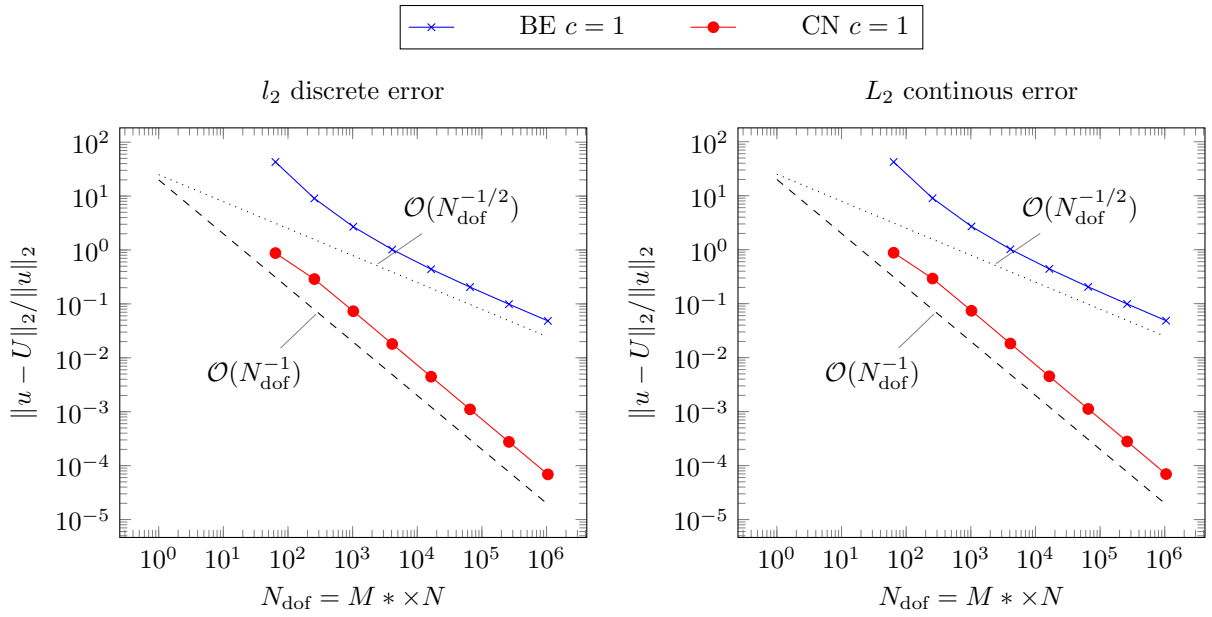


(a) Convergence plots from refinement in the spatial direction.

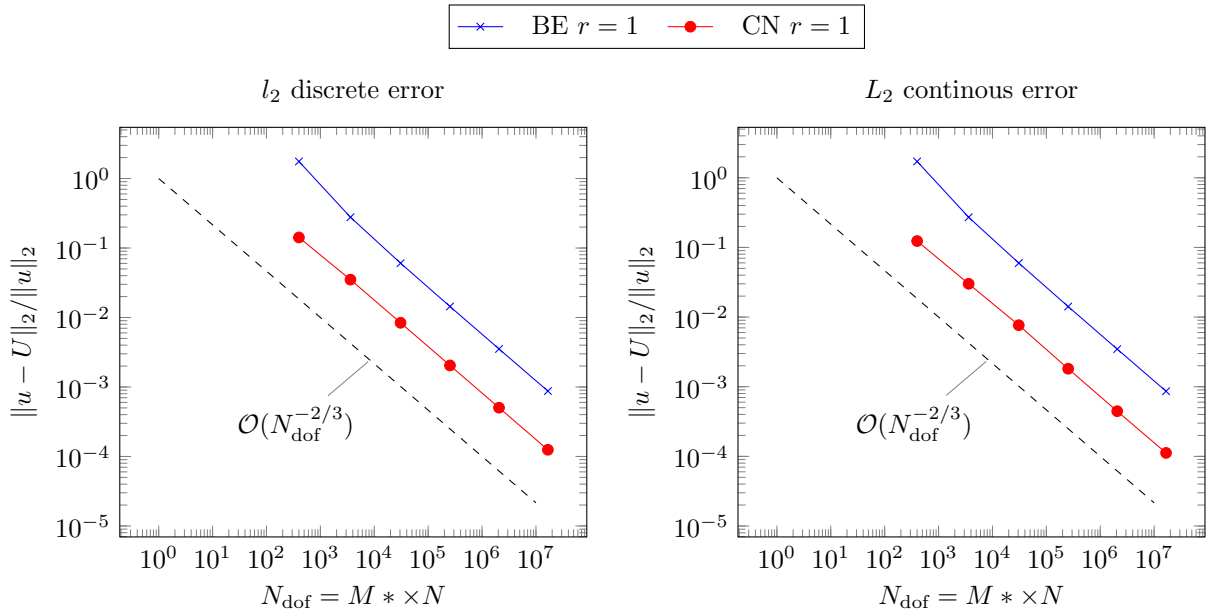


(b) Convergence plots from refinement in the time direction.

Figure 7: Convergence plots from uniform mesh refinement. In 7a we refine in the spatial direction while keeping the number of time steps N constant, then in 7b we refine in the temporal direction while keeping the number of spatial nodes M constant. In both cases both the l_2 discrete and the L_2 continous relative errors is computed. The equation solved is the one-dimensional heat equation $u_t = u_{xx}$ with a sinusoidal initial temperature distribution and the temperature kept fixed at 0 on the boundaries, corresponding to the conditions listed in equation (12).



(a) Convergence plots from uniform refinement with constant $c = k/h$.



(b) Convergence plots from uniform refinement with constant $r = k/h^2$.

Figure 8: Convergence plots from uniform mesh refinement in both the x- and t-direction simultaneously. In 8a $c = k/h$ is kept constant, and in 8b it is $r = k/h^2$ that is kept constant. In both cases both the l_2 discrete and the L_2 continous relative errors is computed. The equation solved is the one-dimensional heat equation $u_t = u_{xx}$ with a sinusoidal initial temperature distribution and the temperature kept fixed at 0 on the boundaries, corresponding to the conditions listed in equation (12).

We see from the convergence plots in figure 8a and figure 8b, that our solver gives the expected convergence. The take home point from here is that the order of convergence when doing simultaneous refinement in both directions depends on the refinement strategy. We see that keeping $c = k/h$ constant is appropriate for Crank-Nicolson, which has the same order of convergence in both h and k, while being suboptimal for Backward Euler which is $\mathcal{O}(k + h^2)$. On the other hand, keeping $r = k/h^2$ works well for Backward Euler, while being suboptimal for Crank-Nicolson, and results in the same convergence order for the two methods, despite Crank-Nicolson having better precision in the time step k.

3 Inviscid Burgers' equation

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

$$u_t = -uu_x, \quad u(0,t) = u(1,t) = 0, \quad u(x,0) = \exp\left(-400(x-1/2)^2\right). \tag{14}$$

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* [4]. The time t_b before this happens can be found exactly using the method of characteristics, and is given as

$$t_b = \frac{-1}{\min f'(x)}, \tag{15}$$

where $f(x)$ is the given initial condition $u(x,0) = f(x)$ [4].

Numerical solution method

To solve equation (14) 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

$$\frac{\partial v_m}{\partial t} = -v_m \frac{1}{2h}(v_{m+1} - v_{m-1}),$$

which we, after imposing the boundary conditions, integrate using an explicit Runge-Kutta method of order 4(5).

3.1 Time of breaking

Insertion of $u(x,0)$ for f in (15), gives $t_b \approx 0.058$. As 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, or equivalently, when there exists a point U_m such that $U_m < U_{m-1}$ and $U_m < U_{m+1}$, we say that the numerical solution has broken down. The time for which this happened for our solution was at $t^* \approx 0.055$, and figure 9 shows the solution sampled around the time of breaking.

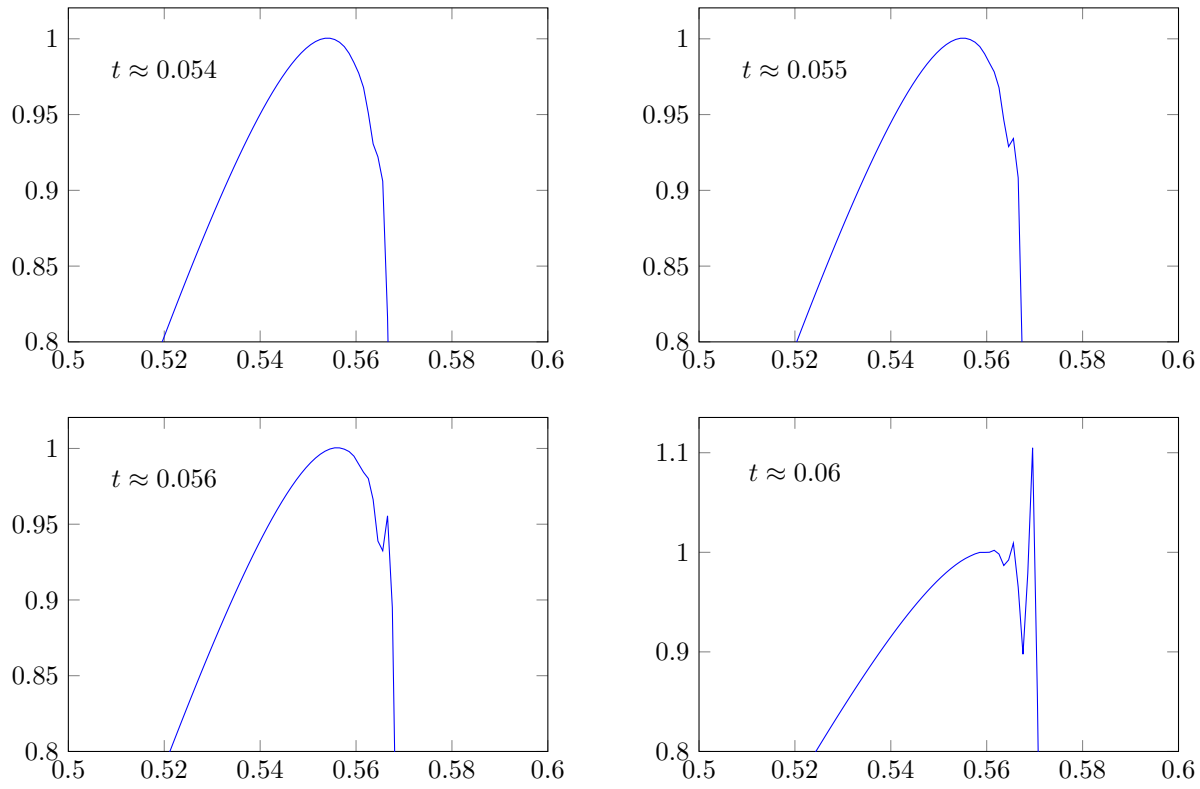


Figure 9: Numerically computed solution to the inviscid Burgers' equation around the time of breaking, displaying the shock formation. The initial condition is $u(x,0) = \exp\left(-400(x-1/2)^2\right)$, and the boundaries are fixed at 0 by the Dirichlet conditions, as specified in (14). The upper right plot with $t \approx 0.055$ is at the time of breaking.

4 Laplace equation in two dimensions

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

$$u_{xx} + u_{yy} = 0,\ (x, y) \in \Omega := [0, 1]^2, \tag{16}$$

with boundary conditions on the edges of Ω

$$\begin{aligned} u(0, y) &= 0, \\ u(1, y) &= 0, \\ u(x, 0) &= 0, \\ u(x, 1) &= \sin(2\pi x). \end{aligned} \tag{17}$$

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 16 can be found by separation of variables. First, asssume 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 indepentent variables. Thus, we have two second order differential equations

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

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 $c = 4\pi$ and the boundary condtitions $\beta(0) = 0$ 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

We solve the equation numerically by discretizing the domain $\Omega = [0,1]^2$, approximate the equation on that domain using a five point stencil, and solving the approximated system. The domain is discretized with $M+2$ and $N+2$ points in the x and y direction, so that there are M and N internal points in each direction. The total system to be solved is thus $M \times N$ points, as the boundaries are known.

Rewriting Laplace's equation using central differences, we get

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

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

where (x_m, y_n) denote the point (m, n) in the grid. Adding these expressions, and naming our approximated solution with the shorthand notation $U_m^n := u(x_m, y_n)$, 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 10,

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

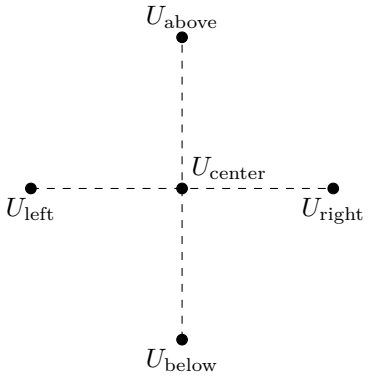


Figure 10: The five-point stencil corresponding to central difference differentiation in both the x- and y-direction. In order to make this more concrete, one can imagine that this stencil is inserted into any point inside a grid such as the one in figure 4. Repeating this process will yield equations for all nodes in the grid, resulting in a solvable system of equations.

We will now construct the matrix A such that we can write our equation as the matrix equation $AU = b$, where U is the flattened solution, and $b = \vec{b}$ contains the boundary conditions of the system, which will be explained in more detail below. 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.

$$A'U = \frac{1}{h^2}\begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix}\begin{bmatrix} U_1^n \\ U_2^n \\ \vdots \\ U_{M-1}^n \\ U_M^n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{M-1}^n \\ b_M^n \end{bmatrix},$$

Note also that this equation only considers one particular value of y, corresponding to n. The boundary conditions on the right hand side are zero for all internal points, while the values along the edges, that is $n = 1, N$ or $m = 1, M$, are set according to (17).

In order to actually solve our entire system, we must include the nodes above and below the center 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 elements $U_1^1, \ldots, U_M^1$, followed by $U_1^2, \ldots U_M^2$ and so on. In this formulation of the problem, the values U_{right} and U_{left} correspond to the neighbouring points in U. The above and below nodes – instead of being above and below U_{center} – are now to the sides, M nodes away, as illustrated in figure 11.

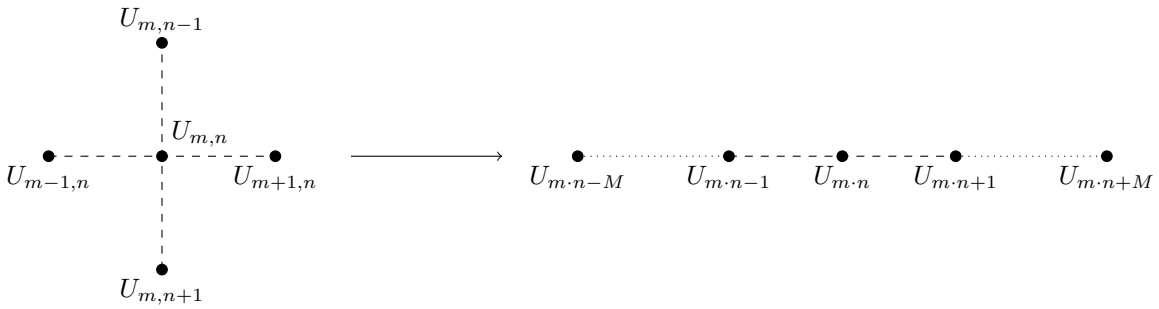


Figure 11: By flattening the five-point stencil, we can write the system of equations, which is then on the form $AU = 0$.

We thus write

$$AU = \begin{bmatrix} \frac{-2}{h^2}+\frac{-2}{k^2} & \frac{1}{h^2} & & \frac{1}{k^2} & & & \\ \frac{1}{h^2} & \frac{-2}{h^2}+\frac{-2}{k^2} & \frac{1}{h^2} & & \frac{1}{k^2} & & \\ & \frac{1}{h^2} & \frac{-2}{h^2}+\frac{-2}{k^2} & 0 & & \frac{1}{k^2} & \\ & \ddots & \ddots & \ddots & & & \\ \frac{1}{k^2} & & 0 & \frac{-2}{h^2}+\frac{-2}{k^2} & \frac{1}{h^2} & & \\ & \frac{1}{k^2} & & \frac{1}{h^2} & \frac{-2}{h^2}+\frac{-2}{k^2} & \frac{1}{h^2} & \\ & & \frac{1}{k^2} & & \frac{1}{h^2} & \frac{-2}{h^2}+\frac{-2}{k^2} & \end{bmatrix}\begin{bmatrix} U_1 \\ \vdots \\ U_{\text{m}} \\ \vdots \\ U_{N\times m} \\ \vdots \\ U_{N\times M} \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \\ \vdots \\ b_{N\times m} \\ \vdots \\ b_{N\times M} \end{bmatrix} = b,$$

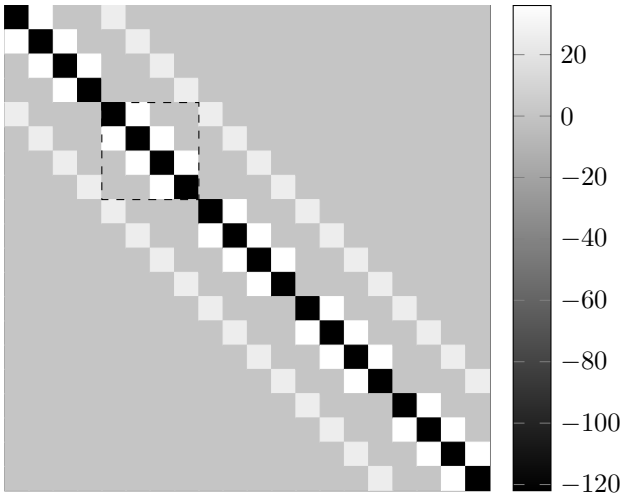


Figure 12: The five point stencil matrix for the case $M = 4$, $N = 5$. Notice the recursive structure, and that the block indicated with a dashed line is repeated $N = 5$ times, while the block itself consist of $M = 4$ elements on the diagonal. Also note the fact that some elements along the first off-diagonals are zero. These elements correspond to nodes along the edge of the system.

which can be solved. Note that the matrix is *not* Toeplitz! There are zeros on the upper and lower diagonal, correponding to the nodes that have less than four neighbours, ie. the nodes on the border. These nodes are handled with the boundary conditions, which come from b. As was mentioned briefly above, the values in b are set in points that correspond to the borders of the system. In this way, the edges of the system are determined with the boundary conditions. The final equation will be on the form $AU = b$, where b and U are flattened matrices, i.e. vectors, of length $N \times M$, while A is a matrix of size $(N \times M)^2$

The large matrix A is also showed in a more managable way in figure 12, where it is plotted as a heatmap. By noticing its recursive structure, one may realize that the matrix can be constructed by a Kronecker sum. This procedure is discussed in depth in **??**, where we show that the stencil can be represented by the Kroenecker sum **??**.

Using the method described above, the solution to equation 16 has been computed, and the results are shown in figure **??**.

Now, as before, we want to perform uniform mesh refinement to analyze the convergence of the difference scheme. To find the expected convergence rate we first find the local error by following the same procedure of inserting the analytical solution into the difference scheme, and taking the arising discrepancy term as the local truncation error τ_m^n. Then we Taylor expand τ_m^n, and we do this now using the computer algebra tool *sagemath*. The result is

$$\begin{aligned}\tau_m^n &= \frac{1}{12}h^2\partial_x^4 u_m^n + \frac{1}{12}k^2\partial_y^4 u_m^n + \mathcal{O}(k^4 + h^4) \\ &= \mathcal{O}(k^2 + h^2).\end{aligned}$$

With this we are ready perform an error analysis by refinement of the grid resolutions in both the x- and the y-direction. We start by refining solely in the x-direction, varying M and keeping N constant, then we switch it around, refining the in the y-direction, i.e. keep M constant while varying N. Finally we do simultaneous refinement in both directions, keeping $c = N/M$ constant, and specifically we set $c = 1$ so that $N = M$.