INF5620 - Numerical methods for partial differential equations

Mandatory Exercise 2

Even Marius Nordhagen

September 12, 2017

- For the Github repository containing programs and results, follow this link: `https://github.com/UiO-INF5620/INF5620-evenmn/tree/master/assignment_2`

1 Introduction

In this project we solve the wave equation with changing phase velocity numerically using Neumann condition. The wave equation is a Partial Differential Equation (PDE) described by

$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial}{\partial x}\left(q(x)\frac{\partial u}{\partial x}\right) + f(x,t) \tag{1}$$

where b is a constant and q is the velocity. The Neumann condition can be implemented in multiple ways. In this project we take a closer look at 4 of them, and then we compare the numerical solutions to the solution

$$u_e(x,t) = \cos(\pi x/L)\cos(\omega t). \tag{2}$$

which can be considered as the exact solution after computing $f(x,t)$. A closer description of the methods are found in the section called "Theory and methods" section. Thereafter we compute the error. The results and conclusion are found in the respective sections.

2 Theory and methods

We have the PDE problem

$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial}{\partial x}\left(q(x)\frac{\partial u}{\partial x}\right) + f(x,t)$$

which can be solved for a given $f(x,t)$ and $q(x)$. To solve this numerically we need to discretize it using the second derivative approximation and the centred scheme, and we get the formula

$$\begin{aligned} u_i^{n+1} = -u_i^{n-1} + 2u_i^n + \frac{1}{2}\left(\frac{\Delta t}{\Delta x}\right)^2 \Big((q_i + q_{i+1})(u_{i+1}^n - u_i^n) \\ -(q_i + q_{i-1})(u_i^n - u_{i-1}^n)\Big) + \Delta t^2 f_i^n \end{aligned} \tag{3}$$

where an arithmetic approximation of the midpoints $q_{i+1/2}$ and $q_{i-1/2}$ is used. You might notice that we need 4 points to calculate a new point, including points from the two previous time points, which is not possible for neither $t = 0$ nor $t = 1$. To solve this problem we need a set of initial conditions.

2.1 Initial conditions

Initial conditions are either given or you have to find them yourself in a clever way. In our case the exact solution is given, so we can use this to find the initial conditions when $t = 0$. We also observe that

$$\frac{\partial u_e}{\partial t} = 0 \quad \text{for} \quad t = 0 \tag{4}$$

which gives us $u_i^1 \approx u_i^{-1}$ using the centred scheme. This makes it possible to find a dedicated formula for the $t = 1$-case:

$$\begin{aligned} u_i^{n+1} = u_i^n + \frac{1}{4}\left(\frac{\Delta t}{\Delta x}\right)^2 \Big((q_i + q_{i+1})(u_{i+1}^n - u_i^n) \\ -(q_i + q_{i-1})(u_i^n - u_{i-1}^n)\Big) + \frac{1}{2}\Delta t^2 f_i^n. \end{aligned} \tag{5}$$

The formula above requires the points u_{i+1} and u_{i-1} when we want to find u_i, so it is useless on the edges. To calculate the points on the edges ($x = 0$ and $x = Nx$ where Nx in the number of spatial points) we need to introduce a boundary condition, and for this project the Neumann condition is appropriate.

2.2 Neumann condition

The Neumann boundary condition specifies the values of the derivative in the boundary points, in our case

$$\frac{\partial u}{\partial x} = 0 \quad \text{for} \quad x = 0, Nx.$$

Similar as for t (see equation (4)), this leads to $u_1^n = u_{-1}^n$, which gives

$$u_i^{n+1} = -u_i^{n-1} + 2u_i^n + \left(\frac{\Delta t}{\Delta x}\right)^2 (q_{i+1/2} + q_{i-1/2})(u_{i\pm1}^n - u_i^n) + \Delta t^2 f_i^n. \quad (6)$$

2.2.1 Method A

The first method, related to subproblem a), is to use the approximation

$$q_{i+1/2} + q_{i-1/2} \approx 2q_i$$

and we obtain

$$u_i^{n+1} = -u_i^{n-1} + 2u_i^n + 2\left(\frac{\Delta t}{\Delta x}\right)^2 q_i(u_{i\pm1}^n - u_i^n) + \Delta t^2 f_i^n \quad (7)$$

for $x = 0, Nx$ and $t = [2, Nt]$ where Nt is the number of time points. Again use equation (4) to find the equation when $t = 1$:

$$u_i^1 = u_i^0 + \left(\frac{\Delta t}{\Delta x}\right)^2 q_i(u_{i\pm1}^0 - u_i^0) + \frac{1}{2}\Delta t^2 f_i^0 \quad (8)$$

2.2.2 Method B

In the second method, related to subproblem b), we use the approximation

$$q_{i+1/2}(u_{i\pm1}^n - u_i^n) - q_{i-1/2}(u_i^n - u_{i\pm1}^n) \approx 2q_{i\pm1/2}(u_{i\pm1}^n - u_i^n)$$

where we assume $dq/dx = 0$. We get the equations:

$$u_i^{n+1} = -u_i^{n-1} + 2u_i^n + 2\left(\frac{\Delta t}{\Delta x}\right)^2 q_{i\pm1/2}(u_{i\pm1}^n - u_i^n) + \Delta t^2 f_i^n \quad (9)$$

for $x = 0, Nx$ and $t = [2, Nt]$, and

$$u_i^1 = u_i^0 + \left(\frac{\Delta t}{\Delta x}\right)^2 q_{i\pm1/2}(u_{i\pm1}^0 - u_i^0) + \frac{1}{2}\Delta t^2 f_i^0 \quad (10)$$

for $x = 0, Nx$ and $t = 1$.

2.2.3 Method C

We are now studying the primitive approximations

$$u_i - u_{i-1} = 0 \quad \text{at} \quad i = 0$$

$$u_{i+1} - u_i = 0 \quad \text{at} \quad i = Nx$$

as described in subproblem c). Apply the arithmetic mean on equation (6) ($q_{i+1/2} \approx 1/2(q_i + q_{i+1})$), and insert the equations above. We then get

$$u_i^{n+1} = -u_i^{n-1} + 2u_i^n + \frac{1}{2}\left(\frac{\Delta t}{\Delta x}\right)^2 (q_i + q_{i\pm1})(u_{i\pm1}^n - u_i^n) + \Delta t^2 f_i^n \tag{11}$$

for $x = 0, Nx$ and $t = [2, Nt]$, and

$$u_i^1 = u_i^0 + \frac{1}{4}\left(\frac{\Delta t}{\Delta x}\right)^2 (q_i + q_{i\pm1})(u_{i\pm1}^0 - u_i^0) + \frac{1}{2}\Delta t^2 f_i^0 \tag{12}$$

for for $x = 0, Nx$ and $t = 1$.

2.2.4 Method D

The last method is about placing a boundary at $x_{1/2}$ and $x_{Nx-1/2}$ by using the formula

$$\frac{\partial^2 u_i^n}{\partial t^2} = \frac{1}{\Delta x}\left(q_{i+1/2}\frac{\partial u_{i+1/2}^n}{\partial x} - q_{i-1/2}\frac{\partial u_{i-1/2}^n}{\partial x}\right) + f_i^n \tag{13}$$

with

$$q_{i+1/2}\frac{\partial u_{i+1/2}^n}{\partial x} = 0 \quad \text{at} \quad x = Nx \quad \text{and} \quad q_{i-1/2}\frac{\partial u_{i-1/2}^n}{\partial x} = 0 \quad \text{at} \quad x = 0.$$

We can discretize using the centred scheme and the arithmetic mean:

$$\frac{\partial u_{i+1/2}^n}{\partial x} \approx \frac{u_i + 1^n - u_i^n}{\Delta x} \quad \text{v} \quad \frac{1}{2}(q_i + q_{i+1})$$

and similar for $\partial u_{i-1/2}^n/\partial x$ and $q_{i-1/2}$. This gives

$$u_i^{n+1} = -u_i^{n-1} + 2u_i^n + \frac{1}{2}\left(\frac{\Delta t}{\Delta x}\right)^2 (q_i + q_{i\pm1})(u_{i\pm1}^n - u_i^n) + \Delta t^2 f_i^n \tag{14}$$

for $x = 0, Nx$ and $t = [2, Nt]$, and

$$u_i^1 = u_i^0 + \frac{1}{4}\left(\frac{\Delta t}{\Delta x}\right)^2 (q_i + q_{i\pm1})(u_{i\pm1}^0 - u_i^0) + \frac{1}{2}\Delta t^2 f_i^0 \tag{15}$$

for for $x = 0, Nx$ and $t = 1$.

2.3 Solve problem

We need to know what $f(x,t)$ is, and we can easily express it by inserting the exact solution to the wave equation. We obtain

$$\begin{aligned} f(x,t) = &\left(q(x)\left(\frac{\pi}{L}\right)^2 - \omega^2 \right) \cos(\pi x/L)\cos(\omega t) \\ &+ \left(\frac{\partial q(x)}{\partial x}\right)\left(\frac{\pi}{L}\right)\sin(\pi x/L)\cos(\omega t) \end{aligned} \tag{16}$$

The remaining part is to take a closer look at $q(x)$, which is the (phase) velocity of the wave. q can be an arbitrary function of x, and in method A it's set to

$$q(x) = 1 + \left(x - \frac{L}{2}\right)^4. \tag{17}$$

In method B we are told to use

$$q(x) = 1 + \cos(\pi x/L) \tag{18}$$

and in method C and D we can choose which $q(x)$ we want to use ourself, but I in this project I have consistent used the first.

We now have a complete set of equations to solve the PDE problem, but there are still a few things that can go wrong. Firstly we need a stable solution, which is the case when Courant number is smaller than one:

$$C = c\frac{\Delta t}{\Delta x} < 1. \tag{19}$$

Secondly we need to implement the equations correctly, briefly explained solve separately for each t and calculate the midpoints before the edge points (see the Github repository for exact implementation).

2.4 Error estimation

The error can be calculated and estimated in multiple ways, and in this problem I decided to find the mean absolute error. Since we have an exact solution, we can easily calculate the absolute error in each spatial point by $E_i = |ue_i - u_i|$ and a fixed T. The total absolute error is the sum of all E_i's, so the mean absolute error is given by

$$\langle E \rangle = \frac{\sum_{i=0}^{Nx} E_i}{Nx} \tag{20}$$

3 Results

3.1 Numerical vs. classical

Since we are given an exact solution, it can be both interesting and useful to plot the exact and the numerical solution in the same figure. A figure for each method is found in figure (1), (2), (3) and (4). To make the methods comparable, we have used $dt = 0.001$, $dx = 0.01$, $T = 0.9$ and all other parameters are set to 1 inn all the 4 figures.

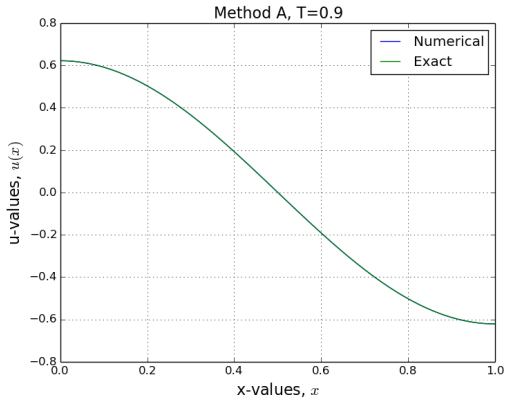

Figure 1: Method A

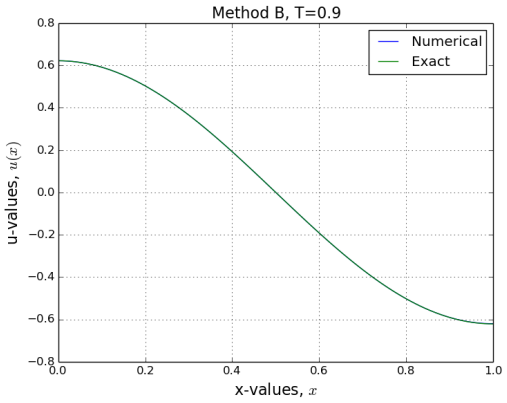

Figure 2: Method B

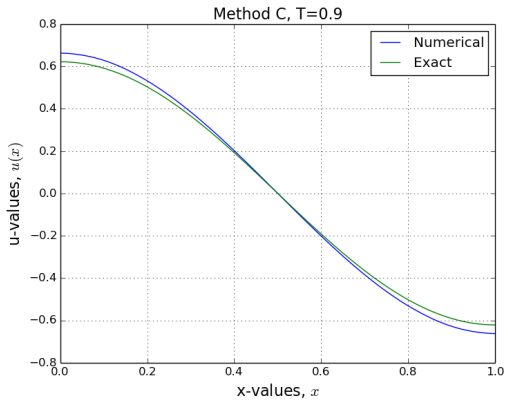

Figure 3: Method C

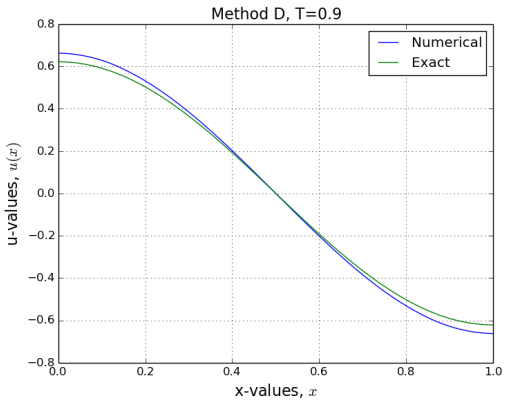

Figure 4: Method D

3.2 Benchmarks

We have now seen how it looks like when we plot the exact solution together with the numerical solutions, thus we are ready to do the benchmark test to see how good the methods really are. We choose $L = 1$ and $dx = 0.01$, so we get 100 spatial points. Due to Courant number, we then need $dt < 0.001$ to get a stable solution, see table (1) for the numerical error values and figure (5) for the corresponding plot.

Table 1: In this table you can find the mean error of the methods where $dt \in [0.001, 0.001, 0.0001]$. $T = 0.9$, all other parameters set to 1.

Methods	$dt = 0.001$	$dt = 0.0001$	$dt = 0.00001$
A	7.207809809e-05	7.067358205e-05	7.053349755e-05
B	0.0001344343021	0.0001427880242	0.0001434920618
C	0.02330151372	0.02331403437	0.02331528300
D	0.02330151372	0.02331403437	0.02331528300

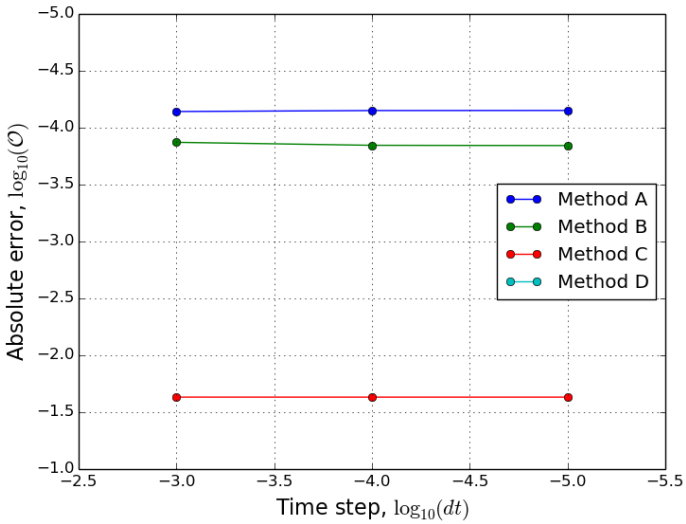

Figure 5: This figure is connected to table (1) and shows the mean error of the methods where $dt \in [0.001, 0.001, 0.0001]$. $T = 0.9$, all other parameters set to 1.

4 Conclusion

From the result we can see that both method A and method B are really good approximations, but this is not the case for method C and D. Because of my choice of approximation method in D, I ended up with exact the same formulas as in C, which might be incorrect. The error is also roughly the same for different dt's, which is not what we expect. Apart from that I am satisfied with my results, the implementation seems to be correct, but perhaps the error calculations are wrong.