Compulsory assignment 2
-INF5620-

Håkon Vikør Treider

Compare discretizations of a Neumann condition

September 28, 2015

Introduction

In this assignment I am going to implement (and test) different ways to discretize the Neumann boundary condition, i.e. $du/dx = 0$ for $x = 0, L$. The code is based on professor H. P. Langtangens code "wave1D_dn_vc.py", which can be found here.

A complication in this assignment is that we allow the wave velocity to be a varying function in space, $q(x)$. To test the convergence rates, we will be using a manufactured solution $u(x,t)$, with a source term $f(x,t)$ to compensate. The source term is calculated in sympy and returned as a valid python function to be used in the solver.

The code for this project can be found on my github-domain, here. The usage of my program is as follows:

```
>> python Neumann_discr.py c 400
```

This example will run task (c) with Nx-values ranging from 50 up to 400 with an spacing of 50 and then print out the computed convergence rates r_i along with some other necessary information (like the size of the timestep etc.). You are then asked if you want to animate a specific value for Nx. Press enter (or 'y' for yes) and then the value you want to test.

The output of convergence rates can be found in the file "results.txt" if you don't have time to wait for the program to finish

Task a)

In this first task, we have $q(x) = 1 + (x - L/2)^4$ which is almost constant, except near the endpoints. This can be seen in figure 1.
On the border we use the central difference to find $u_{n-1} = u_{n+1}$ and also assume $q_{i+\frac{1}{2}1} + q_{i+\frac{1}{2}} \approx 2q_i$ to get

$$u_0^{n+1} = -u_0^{n-1} + 2u_0^n + (\frac{\Delta t}{\Delta x})^2\, 2q_0(u_1^n - u_0^n) + \Delta t^2 f_0^n \tag{1}$$

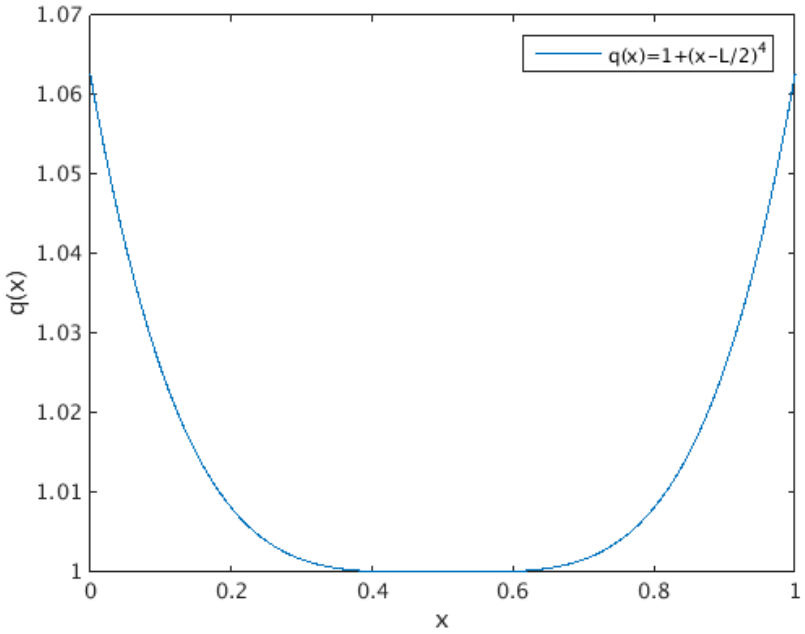

Figure 1: Varying wave velocity $q(x)$ in interval x=0 to x=L=1

with similar results for $i = Nx$.

To find the convergence rate, I compute the difference $e_i^n = u_{i,exact}^n - u_{i,num}^n$ between the numerical solution and the exact manufactured solution at each timestep and save these in an array. I then compute an estimate of the error at each timestep $E_i^n = \sqrt{\Delta t \sum (e_i^n)^2}$ Then at the end of the simulation I pick out the maximum value from this list E and use this as a worst-case-scenario for this run. Then I re-run the simulation with different spatial and temoral resolution. After all the different runs are done, I get the convergence rates from the formula:

$$r_{i-1} = \frac{\ln\left(\frac{E_{i-1}}{E_i}\right)}{\ln\left(\frac{\Delta t_{i-1}}{\Delta t_i}\right)} \tag{2}$$

We can see from output of the program (run with Nx = 500 or higher) , that the convergence rate goes to approximately 2 when we increase the resolution in time and space. This holds true also if we use the other way to discretize the boundary condition, as was already implemented in the program by prof. Langtangen. This is done by switching the variable *Neumann_ver* in the program between "True" and "False".

Task b)

We will now use a variable wave velocity with larger variations in space, $q(x) = 1+\cos(\pi \frac{x}{L})$. Another thing to notice is that $dq/dq = 0$ for $x = 0, L$. This means that the approximation used earlier, $(q_{i+\frac{1}{2}1} + q_{i+\frac{1}{2}} \approx 2q_i)$ now is a second order approximation since its actually true at the boundary. A comparison of the two wave velocities can be seen in figure 2.

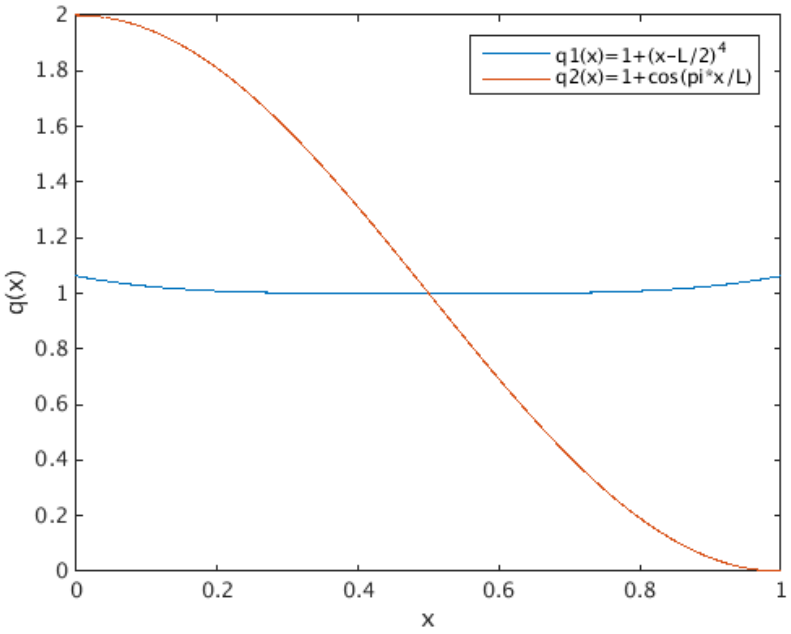

Figure 2: Varying wave velocities $q1(x)$ and $q2(x)$ in interval x=0 to x=L=1

With this new formula for $q(x)$ we can just re-run the program, since we don't need any new implementations. This time we can see a larger difference between the two discretizations. The fact that $dq/dq = 0$ gives the scheme we implemented in task a) a convergence rate of approximately 2.1, meanwhile the standard discretization gives a little less than $r = 2$.

Task c)

In this task we use a more simple way to discretize the Neumann boundary condition, we use the one-sided difference $u_i = u_{i-1}$ when $x = Nx$ and $u_{i+1} = u_i$ when $x = 0$. The error of this scheme then scales linearly with Δt on the boundary, which in turn affect the stability of the entire solution over the whole mesh. From the convergence rates we get this confirmed (as $r \approx 1$). *The exact implementation of this scheme can be viewed in the file "wave1D_dn_vc_modified.py".*

Task d)

In the last task, we are asked to implement the use of "ghost cells". This means adding a non-physical point on both sides of the mesh at $x = -\Delta x x/2$ and $x = L + \Delta x/2$. I have done this, and the program gives a convergence rate of approximately 2, which means the error scales as $O(\Delta t^2)$. *The exact implementation of this scheme can be viewed in the file "wave1D_dn_vc_modified.py".*