KTH Engineering Sciences

Computer Exercise 4
Hyperbolic PDEs

In this exercise you shall make some numerical experiments with finite difference methods applied to two different hyperbolic PDE-problems.

Part 1: Model problem

Consider the model problem for a hyperbolic PDE

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0, \qquad a > 0, \qquad 0 \le x \le D, \qquad t > 0,$$

with initial condition $u(x,0) \equiv 0$. The boundary condition at $x = 0$ models a a wave with period time τ entering from the left into the domain. The wave can be either a sine wave,

$$u(0,t) = g_{\sin}(t) := \sin(2\pi t/\tau),$$

or a square wave,

$$u(0,t) = g_{\mathrm{sq}}(t) := \begin{cases} 1, & \ell\tau < t \le (\ell+1/2)\tau, \\ -1, & (\ell+1/2)\tau < t \le (\ell+1)\tau, \end{cases} \qquad \ell = 0,1,2,\ldots$$

(You can code the last one using MATLAB's `square` command as `"square(2*pi*t/tau)"`, or as `"sign(sin(2*pi*t/tau))"`.)

Your task is to implement the upwind, Lax–Friedrichs and Lax–Wendroff methods, and to compare how they perform for the two different boundary conditions $g_{\sin}$ and g_{sq}. Use a uniform grid in space and set $\tau = 2$, $D = 10$ and $a = 2$. For Lax–Wendroff and Lax–Friedrichs you need a *numerical*[1] boundary condition at $x = D$. Use the extrapolation boundary condition described in Edsberg, Section 8.2.6. (If $u_k^n \approx u(x_k, t_n)$ for $k = 1,\ldots,N$, you use $u_N^n = 2u_{N-1}^n - u_{N-2}^n$. OBS! Same time level n for all three terms.)

(a) (Theoretical preliminaries.) Derive the exact solution $u(x,t)$ for the problem. Hint: It has the form $u(x,t) = f(x - at)$ for some function f.

(b) Run the three methods on the time interval $[0,4]$. Use 100 grid points in space and choose a CFL number $\Delta t/\Delta x$ a bit below the stability limit. Present the results in a 2D graph with $u(x,4)$ as a function of x for *all three methods plotted on top of each other in the same figure together with the exact solution*, for easy comparison. Make one such plot for $g_{\sin}$ and one for g_{sq}.

 – Which method(s) give overly smeared numerical solutions? Which method(s) introduce spurious oscillations? When does this happen?

[1]The PDE itself does not have a boundary condition here!

(c) Make two new program file which can generate plots of several solutions at $t = 4$ on top of each other. The first program should compute solutions with different CFL numbers but fixed Δx. The second program should compute solutions with different Δx but fixed CFL number.

It should be easy to specify the different Δx (or N) and CFL numbers[2] as well as method (upwind, Lax–Friedrichs, Lax–Wendroff) and boundary condition (g_{sin}, g_{sq}). Preferably, varying values should be given as a vector and the rest as scalar variables in the beginning of the program. Do not use interactive input.

Make experiments with the program and be prepared to answer questions like the following ones at the examination.

- How do the smearing and spurious oscillations depend on the CFL number $\Delta t/\Delta x$?
- Which method converges fastest?
- When are the methods stable/unstable? Are your experimental results in agreement with the theoretical stability results?
- Which method is best for the sine and the square wave case, respectively?

You may also be asked to run the program for various configurations to illustrate the answers.

General note: The "magic time step" choice $\Delta t = \Delta x/a$ is quite special here, as it gives the *exact* solution. This only happens with very simple PDEs like the present one. With the PDE in part 2 the choice does not give the exact solution, for example.

Part 2: Water waves

Consider a narrow open channel where water flows with a constant velocity. Waves on the water surface can be seen as small perturbations to the general flow. They are then governed by the linearized Saint–Venant equations, which in rescaled form read

$$\begin{aligned} u_t + u_x + \alpha v_x &= 0, \\ v_t + \alpha u_x + v_x &= f. \end{aligned}$$

Here u and v represent the perturbation of the water level and the water velocity respectively. At $t = 0$ there are no perturbations, so

$$u(x, 0) = 0, \qquad v(x, 0) = 0.$$

Moreover, $\alpha = 1/\text{Fr}$, where Fr is the *Froude number* for the flow[3]. We suppose $\text{Fr} = 0.35$. Finally, the dimensionless function $f = f(x, t)$ models an external disturbance of the flow. We assume here that

$$f(x,t) = s(x)r(t), \qquad s(x) = \begin{cases} \sin(20\pi x), & |x| < 1/20, \\ 0, & \text{otherwise}, \end{cases}, \qquad r(t) = \begin{cases} 1, & \sin\left(40\pi t + \frac{\pi}{6}\right) > 0.5, \\ 0, & \text{otherwise}. \end{cases}$$

[2] When varying the CFL number, you do not need to plot the solution at *exactly* $t = 4$. This allows you to choose the CFL number freely.

[3] The Froude number is a dimensionless number that quantifies the ratio between inertial and gravitational forces acting on the flow. For channel flow it is defined as $\text{Fr} = v_0/\sqrt{gd}$ where v_0 [m/s] is the background flow velocity, d [m] is the channel depth and g [m/s^2] is the gravitational acceleration.

This can correspond to someone repeatedly splashing the water surface at $x = 0$. It generates waves that propagate both upstream and downstream. The channel is infinitely long, but for computational purposes we only consider $x \in [L_0, L_1]$, for $L_0 < 0 < L_1$, and apply artificial boundary conditions at $x = L_0, L_1$ (see below).

(a) (Theoretical preliminaries.) Write the system on the standard form

$$\boldsymbol{v}_t + A\boldsymbol{v}_x = \boldsymbol{F}.$$

- What are $\boldsymbol{v}$, A and $\boldsymbol{F}$?
- Verify that the system is hyperbolic.
- Compute the stability limit given by the CFL condition, i.e. in $\Delta t \leq c\Delta x$, what is c?
- We will use $L_0 = -0.4$ and $L_1 = 0.7$. Because of the finite speed of propagation in hyperbolic problems, the waves generated around $x = 0$ do not reach these artificial boundaries until at $t = T^*$. Estimate T^*. Verify that $T^* > 0.15$.

Hints: To answer these questions you need to compute the eigenvalues of A. They correspond to the speed of propagation of the waves. The source $\boldsymbol{F}$ does not affect the classification of the system (e.g. whether it is hyperbolic or not).

(b) Solve the system with upwind and Lax–Wendroff. Use the same kind of discretization as in Part 1, but note that here the approximation $\boldsymbol{v}_k^n$ is a vector in $\mathbb{R}^2$ and that there is now also a source term $\boldsymbol{F}$ in the right hand side. The forms of upwind and Lax–Wendroff for this case are given at the end of this exercise. Use the same extrapolation boundary condition as in Part 1, now at both boundaries $x = L_0 = -0.4$ and $x = L_1 = 0.7$, for both methods

$$\boldsymbol{v}_0^n = 2\boldsymbol{v}_1^n - \boldsymbol{v}_2^n, \qquad \boldsymbol{v}_N^n = 2\boldsymbol{v}_{N-1}^n - \boldsymbol{v}_{N-2}^n.$$

Simulate the waves in the channel for $0 \leq t \leq 0.15$. Since $0.15 < T^*$ the boundary conditions will not influence the solution. The goal is to provide the following plots:

- Plots of u and v as functions of x at the final time $t = 0.15$. The upwind and Lax–Wendroff solutions should be plotted in the same figure (2 figures, one with the two u solutions and one with the two v solutions).
- 3D plots of u as a function of both t and x, computed with upwind and Lax–Wendroff (2 figures, one for each method).

  Note: The same grids should be used for both methods, so that the results can be compared easily. Moreover, as in Part 1, you do not need to plot the solution at *exactly* $t = 0.15$. Therefore you can choose $\Delta t/\Delta x$ freely.

Write the code so that it is easy to generate the plots with different Δt and Δx, and to turn off the 3D plots (which take very long time for fine discretizations). Make experiments and get an understanding of the solution and how it depends on Δt and Δx. Be prepared to run the code and answer questions like:

- In the exact solution u at $t = 0.15$, which wave peaks, if any, have the same height? Do the left-going or the right-going waves have the shortest (spatial) wave length?

  Note: To be able to answer these questions with confidence you need a rather accurate numerical solution.
- Does your code become unstable if the the CFL condition derived in (a) is violated?

- Can you see smearing and/or spurious oscillations in the solutions? Do these errors seem to be consistent with what you saw in Part 1?
- How can you visually verify the two speeds of propagation from the plots?
- Which method is best for this problem?

Also think about (more difficult):

- Is it harder to approximate the waves going left than those going right? Why?
- The Froude number determines the type of the channel flow. When Fr < 1 (as it is here) the flow is called *subcritical*. How would the solution and code change if the flow is instead *supercritical*, where Fr > 1?

Upwind and Lax–Wendroff for systems

For reference, here is the form of the upwind and Lax–Wendroff method for linear hyperbolic systems with a source. In the schemes we use the notation $\lambda = \Delta t/\Delta x$.

Upwind

Define A^+ and A^- as the positive and negative parts of A as follows

$$A = S\Lambda S^{-1}, \qquad A^+ = S\Lambda^+ S^{-1}, \qquad A^- = S\Lambda^- S^{-1},$$

where $\Lambda^\pm$ is diagonal with negative/positive eigenvalues replaced by zero. Then the upwind method is

$$\boldsymbol{v}_j^{n+1} = \boldsymbol{v}_j^n - \lambda A^+(\boldsymbol{v}_j^n - \boldsymbol{v}_{j-1}^n) - \lambda A^-(\boldsymbol{v}_{j+1}^n - \boldsymbol{v}_j^n) + \Delta t\, \boldsymbol{F}_j^n$$

Lax–Wendroff

The Lax–Wendroff method is

$$\begin{aligned}
\boldsymbol{v}_j^{n+1} &= \boldsymbol{v}_j^n - \frac{\lambda}{2}A(\boldsymbol{v}_{j+1}^n - \boldsymbol{v}_{j-1}^n) + \frac{\lambda^2}{2}A^2(\boldsymbol{v}_{j+1}^n - 2\boldsymbol{v}_j^n + \boldsymbol{v}_{j-1}^n) + \Delta t\, \tilde{\boldsymbol{F}}_j^n \\
\tilde{\boldsymbol{F}}_j^n &= \frac{\boldsymbol{F}_j^n + \boldsymbol{F}_j^{n+1}}{2} - \frac{\lambda}{4}A(\boldsymbol{F}_{j+1}^n - \boldsymbol{F}_{j-1}^n).
\end{aligned}$$

Note the more complicated treatment of the source term.

Matlab tips

The source function f can be defined (after setting w) with the command

```
>> f = @(x,t) (sin(40*pi*t+pi/6)>0.5).*(abs(x)<1/20).*sin(20*pi*x);
```

You can compute A^+ and A^- as

```
>> [S L]=eig(A); Lp = L.*(L>0); Lm = L.*(L<0);
>> Am=S*Lm*inv(S); Ap=S*Lp*inv(S);
```