Homework 2: Linear and nonlinear hyperbolic systems
Max. 4 p

Topics
Conservation form; Relation between non-linear and linearized problems; The Shallow-water equations; The Lax-Friedrichs Scheme; Solution by characteristics; Boundary conditions.
Purpose
To get acquainted with elementary properties of and solution schemes for initial-boundary value problems for hyperbolic systems
Instructions
Write a short report with the plots and answers to the questions posed. Make sure the plots are annotated and there is explanation for what they illustrate.

1 The Shallow Water Model

In this exercise we shall investigate the relation between a non-linear problem and the corresponding linearized system. In particular we will see how well linear analysis predicts the behavior of the nonlinear problem.
Shallow water flow over a horizontal bottom is modeled by

$$\begin{aligned} &h_t + (hv)_x = 0, && \text{(conservation of volume)} \\ &v_t + vv_x + gh_x = 0, && \text{(force balance in } x\text{)} \\ &\text{on } (x,t) \in [0,L] \times [0,\infty), \end{aligned} \qquad (1)$$

where h is the water height (depth) and v the velocity of the water. The PDEs are augmented with boundary conditions $v(0) = v(L) = 0$, and initial conditions representing a localized "water hill",

$$\begin{aligned} &h(x,0) = H + \varepsilon e^{-(x-L/2)^2/w^2}, \\ &v(x,0) = 0. \end{aligned} \qquad (2)$$

We shall take L = 10m, H = 1m, and g = 9.61 (m/s^2). The width w of the water hill is 0.4 m and its height ε will be varied.

1.1 Conservation form (0.5 p)

The mass balance equation is in conservation form, but not the momentum. Introduce $m = hv$ as a new variable instead of v and derive the flux function F in

$$m_t + F(m,h)_x = 0 \qquad (3)$$

h and m are the proper quantities that should be conserved, see the discussion in Leveque. Of course, the smooth solutions, wave speeds, etc., are the same no matter how the equations are written. The quasi-linear form (1) is often most convenient for linearization.

1.2 Numerical Solution (1 p)

To begin with, let ε = 0.1 and solve numerically the conservation form equations using the Lax-Friedrichs method. Use ghost cells at the boundaries. Prescribe values there by the

procedure described in Leveque Ch 7 for solid walls. Choose Δt and Δx after making numerical experiments with different discretization parameters. Use Δt/Δx as large as possible, without violating stability.

1. Make plots showing wave propagation and reflections at the boundaries. Compute at least until waves have been reflected at both boundaries and crossed each other, say until $t = 3.5$.
2. Run the program again with larger values of ε (= 0.4, 0.8, 1.2, …). Describe how the solution changes with respect to
   a. Wave shape, amplitude
   b. Wave speed
   c. You may have to adjust the time-step to ensure stability. Why?

   Illustrate these changed with plots.

1.3 Linearization (1p)

Choose a relevant constant state and derive the linearized problem at that state. Don't forget initial and boundary conditions. Show that the linear problem is hyperbolic and compute wave speeds.

1.4 Analysis of Linear Problem (1p)

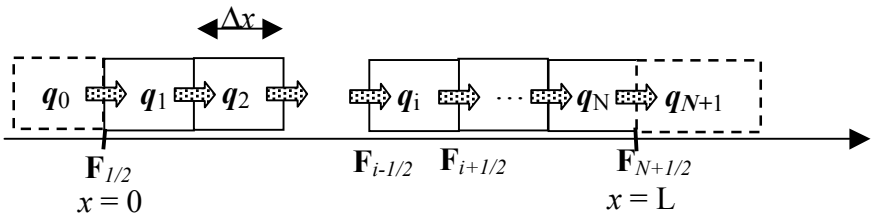
The linear problem can be solved analytically. Determine the solution of the linear problem at for instance $t = 1$. For this you need also the eigenvectors. Discuss how information propagates, if the boundary conditions cause reflections and when reflected waves will appear. Compare with the numerical results for the non-linear case.

1.5 Non-reflecting Boundary Conditions (0.5p)

Derive boundary conditions that *do not* cause reflections for the linear problem. Formulate the corresponding conditions for the non-linear case. Implement the conditions in your program using either the technique of characteristic variables or by simply extrapolating *all* variables at the boundary (as described in Leveque Ch 7). How well does the method work? Try to measure the size of the reflection.

2 Suggestions for program structure

The labs that follow require coding of several finite volume schemes for initial-boundary value problems for a number of different models, mostly explicit. You can of course code as you find practical; we have found the following structure useful. It is possible to separate the scheme from the equation system by defining the *flux function* $\boldsymbol{f}$ as the programming interface between scheme and equation. The state variables $\boldsymbol{q}$ for a system of s equations can be stored in an N x s array, say `Q(1:N,1:s)`



1. Fill ghost cells 0 and N+1 by the boundary conditions; and augment `Q` by rows for the

ghost cells 0 and N+1.

2. Compute the numerical fluxes $\mathbf{F}_{i+1/2}$, $i = 0, 1, \ldots, N$. This may entail much more computation than evaluating the flux function **f** which defines the differential equation, but for the Lax-Friedrichs scheme it is not much more.
3. Compute the flux differences
4. Update the cells $1, \ldots, N$
5. Go to 1

a) For debugging, put plotting into the code so you can inspect the solution and the fluxes and other ingredients *at each time step*; turn off the plotting and printing when the code works.

b) Arrange – by saving solutions on file and implementing interpolation functions, if need be – that solutions from different grids and with different parameter settings can be compared point-wise (in time and space), e.g. for showing convergence in l_2 – norm and showing several solutions in one plot.