In this assignment we are going to implement and test the most basic functionality of our lab-course: The *f-wave* solver for the one-dimensional shallow water equations. The shallow water equations are a system of nonlinear hyperbolic conservations laws with an optional source term:

$$\begin{bmatrix} h \\ hu \end{bmatrix}_t + \begin{bmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \end{bmatrix}_x = S(x,t). \tag{1}$$

The quantities $q = [h, hu]^T$ are defined by $h(x,t)$, the space-time dependent height of the water column and $hu(x,t)$, the space-time dependent momentum in spatial x-direction (u is the particle velocity of the water column). g the gravity constant (usually $g := 9.81m/s^2$) and $f := [hu, hu^2 + \frac{1}{2}gh^2]^T$ the flux function. As source term $S(x,t)$ we will consider the effect of space-dependent bathymetry (topography of the ocean) only $S(x) = [0, -ghB_x]^T$, embedding of additional forces, such as friction or the coriolis effect is possible. Figure 1 illustrates the involved quantities.

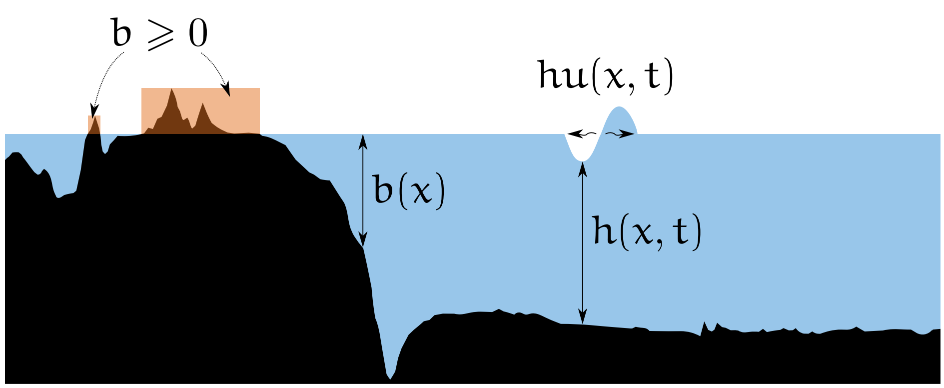

Figure 1: Sketch of quantities appearing in the one-dimensional shallow water equations.

To verify, that this fundamental functionality of our program works as expected, proper (unit-) testing is required. We will do testing by a selection of standardized tests, for which a solution is available.

Remark As units we use meters (m) and seconds (s) for all computations.

Literature

We discuss the basic ideas of numerics, software and strategies in our meetings, nevertheless many important details can't be covered in such a short time. We recommend a basic set of literature in each assignment as hint for your personal studies. In terms of this assignment we recommend the following list of books, papers and guides:

- *Finite volume methods for hyperbolic problems*, R. J. LeVeque, 2002
- *Riemann solvers and numerical methods for fluid dynamics*, E. F. Toro, 2009

- *A wave propagation method for conservation laws and balance laws with spatially varying flux functions*, D. S. Bale et. al., 2003
- *Thinking in C++*, `http://mindview.net/Books/TICPP/ThinkingInCPP2e.html`, Bruce Eckel, 2000
- *git Documentation*: `http://git-scm.com/documentation`
- *Doxygen Manual*: `http://www.stack.nl/~dimitri/doxygen/manual`
- *CxxTest User Guide*: `http://cxxtest.com/guide.html`
- *SCons User Guide*: `http://www.scons.org/doc/production/HTML/scons-user.html`
- *Paraview Documentation*: `http://www.itk.org/Wiki/ParaView/Users_Guide/Table_Of_Contents`

1 The F-wave Solver

In this first chapter we will approximately solve the following Initial Value Problem (IVP) for the shallow water equations (1) over time:

$$q(x, t^n) = \begin{cases} q_l & \text{if } x < 0 \\ q_r & \text{if } x > 0 \end{cases} \qquad q_l, q_r \in \mathbb{R}^+ \times \mathbb{R} \tag{2}$$

Theory shows, that the solution arising from the discontinuity at $x = 0$ consist of two waves, each either a shock or a rarefaction wave. The f-wave solver uses two shock waves to approximate the true solution.

First we use the Roe eigenvalues $\lambda^{\text{Roe}}_{1/2}$ in terms of the left and right quantities q_l and q_r with respect to position $x = 0$ to approximate the true wave speeds:

$$\begin{aligned} \lambda^{\text{Roe}}_1(q_l, q_r) &= u^{\text{Roe}}(q_l, q_r) - \sqrt{g h^{\text{Roe}}(q_l, q_r)} \\ \lambda^{\text{Roe}}_2(q_l, q_r) &= u^{\text{Roe}}(q_l, q_r) + \sqrt{g h^{\text{Roe}}(q_l, q_r)}, \end{aligned} \tag{3}$$

where the height h^{Roe} and particle velocity u^{Roe} are given by:

$$\begin{aligned} h^{\text{Roe}}(q_l, q_r) &= \frac{1}{2}(h_l + h_r) \\ u^{\text{Roe}}(q_l, q_r) &= \frac{u_l\sqrt{h_l} + u_r\sqrt{h_r}}{\sqrt{h_l} + \sqrt{h_r}}. \end{aligned} \tag{4}$$

With the Roe eigenvalues we can define the corresponding eigenvectors $r^{\text{Roe}}_{1/2}$:

$$\begin{aligned} r^{\text{Roe}}_1 &= \begin{bmatrix} 1 \\ \lambda^{\text{Roe}}_1 \end{bmatrix} \\ r^{\text{Roe}}_2 &= \begin{bmatrix} 1 \\ \lambda^{\text{Roe}}_2 \end{bmatrix}. \end{aligned} \tag{5}$$

Decomposition of the jump in the flux function f, $\Delta f := f(q_r) - f(q_l)$, into the eigenvectors gives the waves $Z_{1/2}$:

$$\Delta f = \sum_{p=1}^{2} \alpha_p r_p \equiv \sum_{p=1}^{2} Z_p \qquad \alpha_p \in \mathbb{R}. \tag{6}$$

The left "cell" $\mathcal{C}_l$ is influenced by the left going waves ($\lambda_p < 0$) and the right "cell" $\mathcal{C}_r$ by the right going waves ($\lambda_p > 0$). This leads to the definition of net-updates, which summarize the net-effect of the waves to the left and right "cell":

$$\begin{aligned} A^-\Delta Q &:= \sum_{p:\{\lambda_p^{\mathrm{Roe}}<0\}} Z_p \\ A^+\Delta Q &:= \sum_{p:\{\lambda_p^{\mathrm{Roe}}>0\}} Z_p \end{aligned} \tag{7}$$

Remark You can obtain the eigencoefficients α_p in Equation (6) simply by multiplying the inverse (`http://mathworld.wolfram.com/MatrixInverse.html`) of the matrix of right eigenvectors $R = [r_1^{\mathrm{Roe}}, r_2^{\mathrm{Roe}}]$ to the jump in fluxes:

$$\begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ \lambda_1^{\mathrm{Roe}} & \lambda_2^{\mathrm{Roe}} \end{bmatrix}^{-1} \Delta f. \tag{8}$$

Tasks

1. Create a git-repository for your work. You are free to do this locally or put your code online (i.e. at `https://github.com/`). Use the version control feature of git for all your changes and write meaningful commit-messages. This task is ongoing and has to be fulfilled throughout the complete lab course.

2. Write meaningful Doxygen-documentation for all of your code, especially functions and function-parameters. This task is ongoing and has to be fulfilled throughout the complete lab course.

3. Make extensive use of the `C assert()` macro throughout all of your code. This task is ongoing and has to be fulfilled throughout the complete lab course.

4. Implement the f-wave solver for the homogenous (source term $S(x,t) = 0$) shallow water equations.

   - Input values are the left state $q_l = [h_l, (hu)_l]^T$ and right state $q_r = [h_r, (hu)_r]^T$
   - Output values are
     - Left and right going net-updates: $A^-\Delta Q$ and $A^+\Delta Q$.

– Wave speeds of the left and right going waves: $\lambda_l := \lambda_1^{\mathrm{Roe}}$ and $\lambda_r := \lambda_2^{\mathrm{Roe}}$.
In the case of non-opposite signs appearing in the eigenvalue computation return:

$$\begin{cases} \lambda_l = \lambda_1^{\mathrm{Roe}} \wedge \lambda_r = 0 & \text{if } \lambda_{1/2}^{\mathrm{Roe}} < 0 \\ \lambda_l = 0 \wedge \lambda_r = \lambda_2^{\mathrm{Roe}} & \text{if } \lambda_{1/2}^{\mathrm{Roe}} > 0. \end{cases} \tag{9}$$

5. Write meaningful unit-tests in CxxTest for all implemented functionality. This task is ongoing and has to be fulfilled throughout the complete lab course. Examples for the basic f-wave solver are:

   - Verification of the eigenvalue computation: You can calculate a basic set of eigenvalues for given input values q_l and q_r by using a calculator of your choice.
   - Zero – with respect to machine precision – net-updates in the case of steady states: i.e. $q_l = q_r$.
   - Correctness tests for supersonic problems $\lambda_{1/2}^{\mathrm{Roe}} < 0 \vee \lambda_{1/2}^{\mathrm{Roe}} > 0$: You can derive requirements simply with Eq. (3). Remark: This implies one of the net-updates is zero as stated in Eq. (7).

2 Finite Volume Discretization

During all following tasks of this assignment we assume a finite one-dimensional domain $\Omega := [a, b]$; $a, b \in \mathbb{R}$; $a < b$, which is discretized by n non-overlapping cells $\mathcal{C}_i$, with $\Omega = \cup_{i=1}^{n} \mathcal{C}_i$. In each cell i we specify the space- and time-dependent set of quantities $Q_i = (h, hu)_i$ and the space-dependent bathymetry b_i: Again $h_i \in \mathbb{R}^+$ is the total water height, $(hu)_i \in \mathbb{R}$ the momentum and $b_i \in \mathbb{R}$ the bathymetry relative to the sea level. Additionally we have to specify proper boundary conditions, which is done by the two ghost cells $\mathcal{C}_0$ and $\mathcal{C}_{n+1}$ neighboring the left and right boundary. The obtained spatial discretization is called Finite Volume discretization.

With a given Finite Volume discretization we can define a set of $n+1$ edge-local Riemann problems:

$$q(x, t^n) = \begin{cases} Q_{i-1} & \text{if } x < x_{i-1/2} \\ Q_i & \text{if } x > x_{i-1/2} \end{cases} \qquad \forall i \in \{1, .., n+1\} \tag{10}$$

The solution of these Riemann problems is analogue to the single-Riemann problem case in Eq. (2): Now the f-wave solver determines two net-updates $A^{\mp}\Delta Q_{i-1/2}$ per edge $x_{i-1/2}$. The net-update $A^-\Delta Q_{i-1/2}$ is the net-effect of left going waves and influences the quantities of C_{i-1}. $A^+\Delta Q_{i-1/2}$ summarizes the net-effect of the right going waves and influences the quantities in C_i.

The final update formula from time step t^n to the next time step $t^{n+1} = t^n + \Delta t$ by the time step width $\Delta t \in \mathbb{R}^+$ is given by:

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x}\left(A^+\Delta Q_{i-1/2} + A^-\Delta Q_{i+1/2}\right) \qquad \forall i \in \{1, .., n\}. \tag{11}$$

Thus a cell C_i is influenced by the right going waves – summarized by the net-update $A^+\Delta Q_{i-1/2}$ – of its left edge at position $x_{i-1/2}$ and the left going waves – summarized by the net-update $A^-\Delta Q_{i+1/2}$ – of its right edge at position $x_{i+1/2}$. $\Delta x \in \mathbb{R}^+$ is the cell width, which is constant in our implementations.

Δt is chosen in a way, that the waves do not interact with each other, which is equivalent to waves, which do not cross the cell centers x_i. In that case stability of the method is guaranteed if the time step width Δt is chosen to satisfy the CFL-criterion:

$$\Delta t < \frac{1}{2} \cdot \frac{\Delta x}{\lambda^{\max}} \tag{12}$$

where $\lambda^{\max}$ is maximum over all absolute values of the eigenvalues computed on the $n+1$ edges:

$$\lambda^{\max} = \max_{i=1}^{n+1} \left(\left| \lambda^{\text{Roe}}_{1/2} \right| \right)_{i-1/2} \tag{13}$$

Remark We do not use the bathymetry in this assignment, you can simply set a dummy value in all cells: $b_i = 0 \quad \forall i \in 0 \ldots n+1$. SWE1D comes with outflow boundary conditions, which set appropriate values in the ghost cells $\mathcal{C}_0$ and $\mathcal{C}_{n+1}$ in every time step `https://github.com/TUM-I5/SWE1D/blob/master/src/WavePropagation.cpp#L75`.

We will implement the bathymetry in our f-wave solver and implement new boundary conditions as part of the second assignment.

Tasks

1. Checkout the SWE1D-Code from `https://github.com/TUM-I5/SWE1D`.

2. Make yourself familiar with the SCons software construction tool `http://www.scons.org`. A very basic top-level SConstruct file for SWE1D is located at `https://github.com/TUM-I5/SWE1D/blob/master/SConstruct`.
   Remark CxxTest comes with a build tool for the smooth integration into SCons.

3. Remove the existing symbolic link `https://github.com/TUM-I5/SWE1D/tree/master/src/solvers` and replace it with a symbolic link pointing to a git-submodule containing your f-wave implementation. Integrate your f-wave solver, developed in Task 4 of Chapter 1, into SWE1D. Visualize the pre-defined dam-break scenario: `https://github.com/TUM-I5/SWE1D/blob/master/src/scenarios/dambreak.h`
   Remark Almost every Finite Volume functionality is already provided by SWE1D, thus you have to modify your solver to match with the call in `https://github.com/TUM-I5/SWE1D/blob/master/src/WavePropagation.cpp` only. The following setups in Chapter 3.1, 3.2 and 4 provide a good collection for your unit tests, because analytical solutions of each problem are comparable easy to derive. The csv-file `middle_states.csv` located at `http://www5.in.tum.de/lehre/praktika/swe_lab_ss13/middle_states.csv` contains a collection of constant water heights, which arise immediately in a Riemann problem at the initial discontinuity x_{dis} at time $t > 0$.

3 Shock and rare problems

3.1 Shock-Shock Problem

In this Chapter we will solve "shock-shock"-Riemann problems using SWE1D. You can imagine two streams of water, which move in opposite directions and smash into each other at some position x_{dis}. The scenario is given by the following setup:

$$\begin{cases} Q_i = q_l & \text{if } x_i \leqslant x_{dis} \\ Q_i = q_r & \text{if } x_i > x_{dis} \end{cases} \qquad q_l \in \mathbb{R}^+ \times \mathbb{R}^+,\ q_r \in \mathbb{R}^+ \times \mathbb{R}^- \tag{14}$$

with initial conditions

$$q_l = \begin{bmatrix} h_l \\ (hu)_l \end{bmatrix}, \quad q_r = \begin{bmatrix} h_r \\ (hu)_r \end{bmatrix} = \begin{bmatrix} h_l \\ -(hu)_l \end{bmatrix}. \tag{15}$$

3.2 Rare-Rare Problem

We can setup "rare-rare" Riemann problems by two streams of water, which move away from each other at some position x_{dis}. The scenario is defined as:

$$\begin{cases} Q_i = q_l & \text{if } x_i \leqslant x_{dis} \\ Q_i = q_r & \text{if } x_i > x_{dis} \end{cases} \qquad q_l \in \mathbb{R}^+ \times \mathbb{R}^-,\ q_r \in \mathbb{R}^+ \times \mathbb{R}^+ \tag{16}$$

with initial conditions identical to Equation (15).

Tasks

1. Implement the shock-shock and rare-rare problems as a scenario in SWE1D: `https://github.com/TUM-I5/SWE1D/tree/master/src/scenarios`.
2. Play around with different sets of initial water heights h_l and particles velocities u_l. What do you observe? Is there a connection to the wave speeds $\lambda_{1/2} = u \mp \sqrt{gh}$ of Chapter 1?

4 Dam-Break

In this section we will solve the "dam-break" problem. You can imagine a water reservoir, which is separated as illustrated in Fig. 2 from a river by a dam wall initially. Now we assume a total failure of the dam wall, thus nothing keeps the water from moving down the river:

$$\begin{cases} Q_i = q_l & \text{if } x_i \leqslant x_{dis} \\ Q_i = q_r & \text{if } x_i > x_{dis} \end{cases} \qquad q_l \in \mathbb{R}^+ \times 0,\ q_r \in \mathbb{R}^+ \times \mathbb{R}^+,\ h_l > h_r \tag{17}$$

Figure 3 shows the analytical solution of the dam-break problem, which consists of a rarefaction and a shock wave.

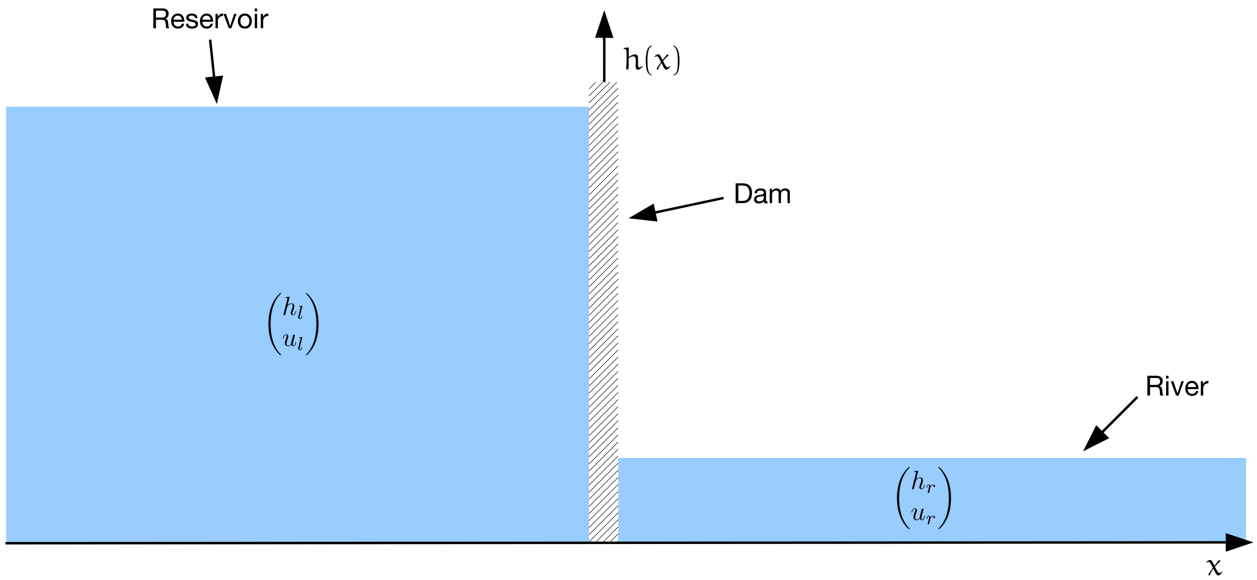

Figure 2: Initial dam break problem.

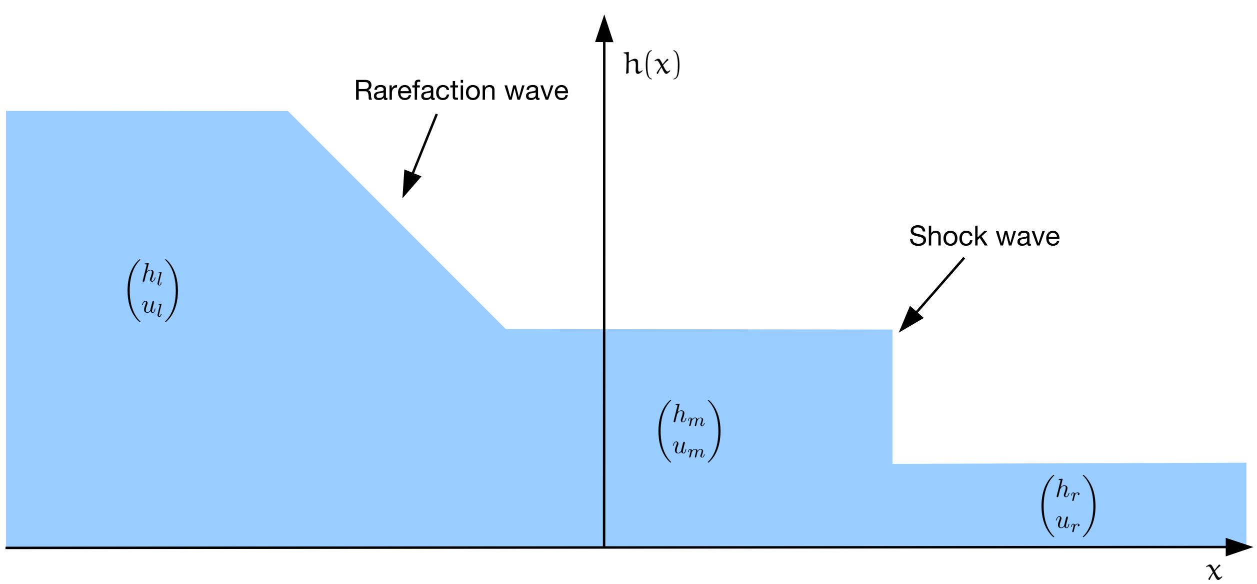

Figure 3: Solution of the dam break problem.

1. Extend the dam-break scenario of SWE1D `https://github.com/TUM-I5/SWE1D/blob/master/src/scenarios/dambreak.h` and play around with different sets of initial water heights h_l and h_r. What do you observe? How large is the impact of the particle velocity u_r in the river?

2. Assume a water reservoir of unlimited size and a village 25 km down the river with initial values $q_l = [14, 0]^T$ and $q_r = [3.5, 0.7]^T$. How much time do you have to evacuate the village in our model before the wave arrives?

Deliverables

The following deliverables have to be handed in no later than 08:00 AM, Monday, 29th April, 2013. Small files (<1 MB in total) can be send as an attachment directly to *breuera AT in.tum.de* and *rettenbs AT in.tum.de*, larger files have to be uploaded at a place of your choice, i.e. `https://github.com/`, `http://home.in.tum.de/`, `https://www.dropbox.com`. In either case inform us about the final state of your solution via e-mail.

- Code in a git-repository. Doxygen documentation and unit tests are mandatory.
- Slides for the presentation during the next meeting.
- Pictures and animations of all runs.
- Documentation how to build and use your code, especially if you extend the SCons script.
- Doxygen documentation as html and pdf.