In their differential form the two dimensional shallow water equations are given by:

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

The quantities $q = [h, hu, hv]^T$ are defined by the space-time dependent water height $h(x, y, t)$, the momentum in spatial x-direction $hu(x, y, t)$ and the momentum in y-direction $hv(x, y, t)$. $F := [hu, hu^2 + \frac{1}{2}gh^2, huv]^T$ is the flux function in x-direction and $G := [hv, huv, hv^2 + \frac{1}{2}gh^2]^T$ in the flux function in y-direction. g is the gravitational constant (usually $g := 9.81\text{m/s}^2$). As for the one-dimensional shallow water equations, we consider only the space-dependent effect of bathymetry in the source term $S(x, y) = [0, -ghB_x, -ghB_y]^T$.

In this assignment we will first introduce bathymetry in our solver and simulate one dimensional flow over a weir in Chapter 1 and 2. The following Chapters 3 and 4 cover the extension of our one dimensional discretization to two spatial dimensions. After this assignment we have all numerics at hand and will be able to concentrate on Tsunami simulations in the third assignment.

Literature

- *Hydraulics over a weir*: `http://serc.carleton.edu/details/files/19076.html`
- *SWE Wiki*: `https://github.com/TUM-I5/SWE/wiki`
- *Documentation, GNU C Preprocessor*: `http://gcc.gnu.org/onlinedocs/cpp/`
- *Case Study: Malpasset Dam-Break Simulation using a Two-Dimensional Finite Volume Method*, Valiani et. al., 2002

1 Bathymetry

1.1 Non-zero source term

Bathymetry is the topography of the ocean and the most important effect to be considered in the source term of the shallow water equations for Tsunami simulations. As reference we use the sea level: Bathymetry in dry cells (land) $\mathcal{C}_i$ is zero or positive: $b_i \geqslant 0$, wet cells (water) have negative values: $b_i < 0$.

Introduction of bathymetry into the f-wave solver is comparably simple, because it can be directly incorporated into the decomposition of the jump in the flux function:

$$\Delta f - \Delta x \Psi_{i-1/2} = \sum_{p=1}^{2} Z_p, \tag{2}$$

where the term $\Delta x \Psi_{i-1/2}$ summarizes[1] the effect of the bathymetry:

$$\Delta x \Psi_{i-1/2} := \begin{bmatrix} 0 \\ -g(b_r - b_l)\frac{h_l + h_r}{2} \end{bmatrix}. \tag{3}$$

1.2 Reflecting boundary conditions

Introduction of bathymetry naturally leads to the question of handling inundation and water, which moves back to sea, in near-shore areas $b(x) \approx 0$. The f-wave solver is not capable of handling wetting and drying. We will therefore impose two simplifications to keep our solver stable:

- Cells $\mathcal{C}_i$, which are initially wet, stay wet during the whole simulation: We assure each cell has enough water initially for the complete time of the simulation. A fix for Tsunami simulations will be introduced as part of the third assignment, where we implement reading of measured bathymetry data.
- Cells $\mathcal{C}_i$, which are initially dry, stay dry during the whole simulation: At a wet-dry interface reflecting boundary conditions are used.

If a wave moves from a wet region towards the shore, we assume an infinitly high solid wall at the shore-line, which is equivalent to reflecting boundary conditions. Thus no water is able to pass the shoreline, and we have to assure that the particle velocity at the boundary is zero. In our Finite Volume discretization we have neighboring cells $\mathcal{C}_{i-1}$ and $\mathcal{C}_i$, where $\mathcal{C}_{i-1}$ is wet and $\mathcal{C}_i$ is dry, Fig. 1 illustrates the setting. As seen in the shock-shock example of

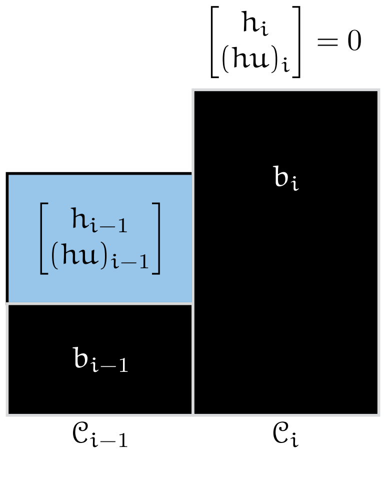
$\begin{bmatrix} h_i \\ (hu)_i \end{bmatrix} = 0$
b_i
$\begin{bmatrix} h_{i-1} \\ (hu)_{i-1} \end{bmatrix}$
b_{i-1}
$\mathcal{C}_{i-1}$
$\mathcal{C}_i$

Figure 1: Wet-dry boundary for a wet cell $\mathcal{C}_{i-1}$ and dry cell $\mathcal{C}_i$.

the first assignment already, we have a middle state with zero particle velocity, if we set the opposite momentum in the dry cell combined with equal water height. Additionally setting

[1]Details can be found in: *A wave propagation method for conservation laws and balance laws with spatially varying flux functions*, D. S. Bale et. al., 2003

the bathymetry in the dry cell to the value of the wet cell $b_i = b_{i-1}$ completes the setup of reflecting boundary conditions:

$$\begin{aligned} h_i &= h_{i-1} \\ (hu)_i &= -(hu)_{i-1} \\ b_i &= b_{i-1} \end{aligned} \tag{4}$$

Tasks

1. Extend the f-wave solver to handle bathymetry according to Equations (2) and (3). Implement a simple example, which shows the effect of bathymetry.
   Remark You have to extend SWE1D to support bathymetry as well. In the ghost cell $b_0 := b_1$ and $b_{n+1} = b_n$ is set initially.

2. Implement reflecting boundary conditions (4) and show that you obtain a one-sided solution of the shock-shock setup from the first assignment, if you use q_l in the complete spatial domain with reflecting boundary conditions at the right boundary and outflow boundary conditions at the left boundary.
   Remark Reflecting boundary conditions for the right boundary means, that the ghost cell $\mathcal{C}_{n+1}$ in the description above is dry.

2 Hydraulic jumps

In this chapter we will compute the behavior of flows in a sub- and supercritical example. In shallow water theory we can characterize this behavior by the Froude number:

$$F := \frac{u}{\sqrt{gh}}, \tag{5}$$

where we call regions with $F < 1$ subcritical, $F \approx 1$ critical and $F > 1$ supercritical, which matches with the wave speeds of the first assignment in terms of eigenvalues $\lambda_{1/2} = u \mp \sqrt{gh}$. In the supercritical case the flow velocity is larger than the wave velocity relative to the fluid $\mp\sqrt{gh}$. Information travels in one direction only and hydraulic jumps can occur. This behavior of flow is closely related to the subsonic, sonic and supersonic cases in acoustics.

The two following examples[2] set up sub- and supercritical flow over a hump. As computational domain we use the interval $(0, 25)$ discretized with cells $\mathcal{C}_i$, $i \in 1 \dots n$, where the ghost cells $\mathcal{C}_0$ and $\mathcal{C}_{n+1}$ correspond to positions $x_0 = 0$ and $x_{n+1} = 25$. We use outflow boundary conditions in both cases.

[2]Details of the original benchmarks can be found in: *Augmented Riemann solvers for the shallow water equations over variable topography with steady states and inundation*, D. L. George, 2008

2.1 Subcritical flow

For the subcritical flow we use the following initial values:

$$\begin{aligned} b(x) &= \begin{cases} -1.8 - 0.05(x-10)^2 & \text{if } x \in (8,12) \\ -2 & \text{else} \end{cases} \\ h(x,0) &= -b(x) \quad \text{if } x \in [0,25] \\ hu(x,0) &= 4.42 \quad \text{if } x \in [0,25]. \end{aligned} \tag{6}$$

2.2 Supercritical flow

For the supercritical flow we use the following initial values:

$$\begin{aligned} b(x) &= \begin{cases} -0.13 - 0.05(x-10)^2 & \text{if } x \in (8,12) \\ -0.33 & \text{else} \end{cases} \\ h(x,0) &= -b(x) \quad \text{if } x \in [0,25] \\ hu(x,0) &= 0.18 \quad \text{if } x \in [0,25]. \end{aligned} \tag{7}$$

Tasks

1. Compute the location and value of the maximum Froude number for the subcritical setting (6) and the supercritical setting (7) at initial time $t = 0$.

2. Implement both examples as a scenario in `https://github.com/TUM-I5/SWE1D/tree/master/src/scenarios`, $t \in [0, 200]$ is reasonable as temporal domain for the simulation.

3. Determine the position of the hydraulic jump (stationary discontinuity) in your supercritical solution and show, that our f-wave solver fails to converge to the analytically expected constant momentum over the complete domain.

3 Dimensional splitting

The method of dimensional splitting allows us to apply our f-wave solver, which is designed to solve the one-dimensional shallow water equations, easily to two spatial dimensions. The idea is to split the two dimensional problem into two one-dimensional ones, which we solve after each other in so called sweeps. The x-sweep in spatial x-direction is given by:

$$Q^*_{i,j} = Q^n_{i,j} - \frac{\Delta t}{\Delta x}\left(A^+\Delta Q_{i-1/2,j} + A^-\Delta Q_{i+1/2,j}\right) \qquad \forall i \in \{1,..,n\},\ j \in \{0,..,n+1\}, \tag{8}$$

the y-sweep in spatial y-direction is applied to the intermediate values $Q^*_{i,j}$ and defined as:

$$Q^{n+1}_{i,j} = Q^*_{i,j} - \frac{\Delta t}{\Delta y}\left(B^+\Delta Q^*_{i,j-1/2} + B^-\Delta Q^*_{i,j+1/2}\right) \qquad \forall i,j \in \{1,..,n\}. \tag{9}$$

As in the one-dimensional case, a cell $\mathcal{C}_i$ is influenced by all neigboring cells, which are connected through the net-updates $A^{\pm}\Delta Q_{i\mp 1/2,j}$ on the vertical edges $x_{i\mp 1/2,j}$ and $B^{\pm}\Delta Q^{*}_{i,j\mp 1/2}$ on the horizontal edges $y_{i,j\mp 1/2}$. The time step restriction due to the CFL criterion for Δt is similar to the one-dimensional case, but has to be true in the x- and y-sweep:

$$\Delta t < \frac{1}{2} \cdot \frac{\Delta x}{\lambda_x^{\max}} \quad \wedge \quad \Delta t < \frac{1}{2} \cdot \frac{\Delta y}{\lambda_y^{\max}}, \tag{10}$$

where $\lambda_x^{\max}$ is the maximum eigenvalue appearing in the x-sweep and $\lambda_y^{\max}$ the maximum eigenvalue in the y-sweep.

Remark We use a slightly pessimistic time step width for both updates, derived in the x-sweep already:

$$\Delta t = 0.4 \cdot \frac{\Delta x}{\lambda_x^{\max}}, \tag{11}$$

which most probably leads to a stable method for squares, $\Delta x = \Delta y$.

Tasks

1. Checkout SWE from `https://github.com/TUM-I5/SWE`. Compile and run the existing code.
   Remark SWE provides several features, such as CUDA or MPI, which are not part of the regular assignments, but could be used in the project phase.

2. Implement the dimensional splitting method with your f-wave solver of the first assignment:

   - Remove the existing solvers symbolic link `https://github.com/TUM-I5/SWE/blob/master/src/solvers` and replace it with a symbolic link pointing to a git-submodule containing your f-wave implementation.
   - Extend the abstract class `SWE_Block` with a class `SWE_DimensionalSplitting`, which implements the dimensional splitting method.
   - Add an optional (compiles only, if the preprocessor variable `NDEBUG` is not defined) debug statement, which checks if the CFL condition (10) in the y-sweep is satisfied.
   - Implement your own dimensional splitting example `swe_dimensionalsplitting.cpp` in `https://github.com/TUM-I5/SWE/tree/master/src/examples`; provide an option in the SCons-script `https://github.com/TUM-I5/SWE/blob/master/SConstruct`, which compiles your implementation.

3. Provide unit tests, which check the two-dimensional implementation in SWE against the implementation in SWE1D.

4 Partial Dam-Break

In this chapter we will simulate the dam break of a water reservoir in two spatial dimensions. The water reservoir has size 97.5m × 200m and is separated from a river by a dam of 5m thickness, which is partially broken at a length of 75m. The simulated domain of the river is 297.5m × 100m. Figure 2 illustrates the geometrical setting. The initial values in the water

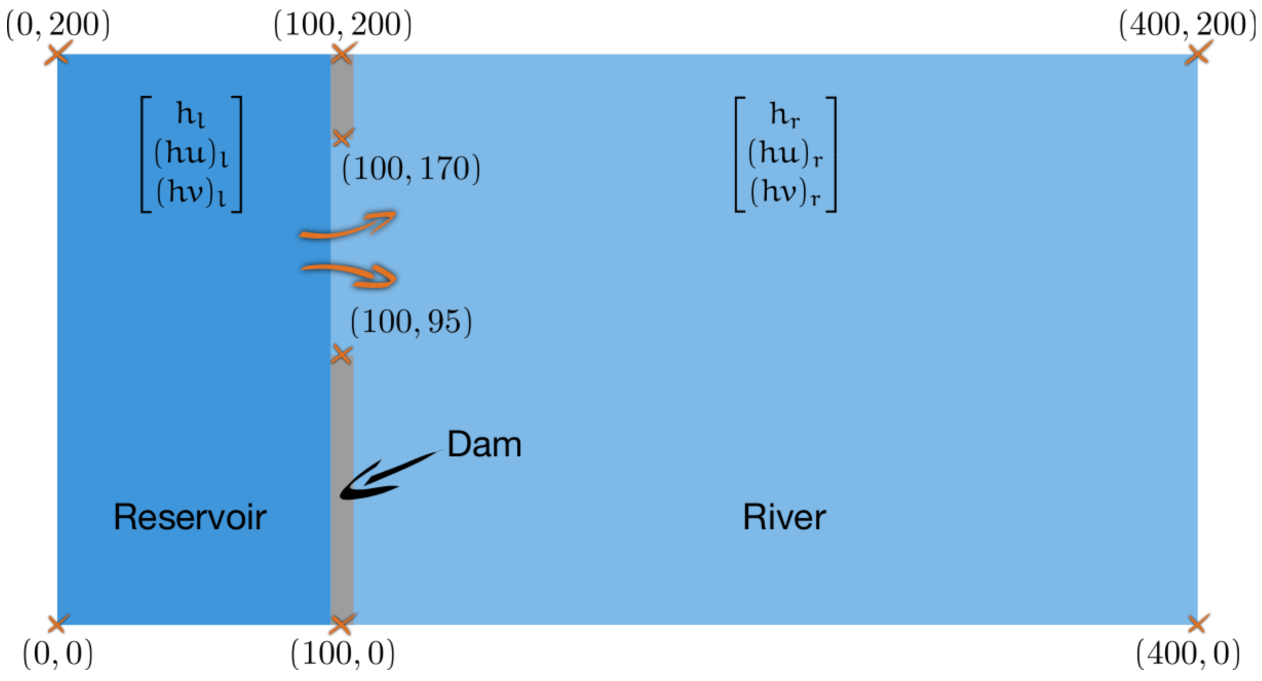
$(0, 200)$
$(100, 200)$
$(400, 200)$
$\begin{bmatrix} h_l \\ (hu)_l \\ (hv)_l \end{bmatrix}$
$(100, 170)$
$\begin{bmatrix} h_r \\ (hu)_r \\ (hv)_r \end{bmatrix}$
$(100, 95)$
Dam
Reservoir
River
$(0, 0)$
$(100, 0)$
$(400, 0)$

Figure 2: Initialization of the partial dam-break problem.

reservoir q_l and the river q_r are given by:

$$q_l = \begin{bmatrix} h_l \\ (hu)_l \\ (hv)_l \end{bmatrix} = \begin{bmatrix} 10 \\ 0 \\ 0 \end{bmatrix}, \qquad q_r = \begin{bmatrix} h_r \\ (hu)_r \\ (hv)_r \end{bmatrix} = \begin{bmatrix} 5 \\ 0 \\ 0 \end{bmatrix}. \tag{12}$$

As boundary conditions we use outflow boundary conditions for the right boundary and reflecting boundary conditions for all other boundaries.

1. Implement the wet-dry handling described in Chapter 1.2 inside your f-wave solver.
2. Implement the partial dam-break setting as a scenario in SWE `https://github.com/TUM-I5/SWE/tree/master/src/scenarios`.
3. Play around with different initial water heights. What do you observe?

Deliverables

The following deliverables have to be handed in no later than 08:00 AM, Monday, 13th May, 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 SConscript.