S. Mishra
L. Scarabosio
J. Šukys

Spring Term 2014

Numerical Methods for Partial Differential Equations

ETH Zürich
D-MATH

Homework Problem Sheet 6

Introduction. This exercise sheet focuses on the stable FVM discretizations of the *non-linear* conservation laws. The first problem is the implementation of various numerical flux functions for the approximate Riemann solver and application of the methods for the one-dimensional Burgers' equation. The second problem shows how to apply many of the FVM schemes that you have already learned to a more practical setting. In particular, you will solve an example of a *non-linear multi-dimensional system* of hyperbolic conservation laws - the so-called shallow water equations. Such equations are used to model dynamics in fluids, where depth is very small, compared to the width and length of a given physical domain. In particular, flows in oceans, rivers, lakes, as well as atmospheric flows of air masses are often modeled using shallow water equations. In this problem, we will use a very simplified model of a "dambreak", in one and two space dimensions, modeling the corresponding physical 2-D and 3-D flows, respectively. Upon multiple requests, the master solution for the second problem will be provided in Python (however, MATLAB users should be able to easily understand the code, as many things are similar).

Problem 6.1 Burgers' equation: Rankine-Hugoniot condition and Riemann solvers (Core problem)

Consider the one-dimensional Burger's equation:

$$U_t + \left(\frac{U^2}{2}\right)_x = 0, \qquad \forall (x,t) \in D \times \mathbb{R}_+, \tag{6.1.1}$$

with domain $D = (-1, 3)$ and following initial conditions:

$$\textbf{(Shock)} \qquad U(x,0) = \begin{cases} 1, & \text{if} \quad x < 0, \\ 0, & \text{otherwise}, \end{cases} \tag{6.1.2}$$

$$\textbf{(Rarefaction)} \qquad U(x,0) = \begin{cases} -1, & \text{if} \quad x < 0, \\ 1, & \text{otherwise}, \end{cases} \tag{6.1.3}$$

$$\textbf{(Rarefaction and Shock)} \qquad U(x,0) = \begin{cases} 1, & \text{if} \quad 0 < x < 1, \\ 0, & \text{otherwise}. \end{cases} \tag{6.1.4}$$

(6.1a) Using *Rankine-Hugoniot conditions* (for shock waves) and *entropy conditions* (to distinguish between shocks and rarefaction waves), derive the expression for *weak entropy* solutions of (6.1.1) and all three initial conditions (6.1.2) - (6.1.4).

(6.1b) Implement a finite volume code for the Burgers' equation (6.1.1), using Godunov's, Roe, Lax-Friedrichs and Rusanov's numerical flux functions. Compute the numerical solution for the initial conditions (6.1.2), (6.1.3), and (6.1.4), using *outflow boundary conditions*. Plot the solutions at time $T = 1.0$ for 100 mesh cells. Explain the observed results.

HINT: *Outflow boundary conditions* can be easily implemented by extrapolating the values of the variable to the ghost cell i.e. $U_0^n = U_1^n$ and $U_{N+1}^n = U_N^n$.

(6.1c) Use the exact solutions derived in sub-problem (6.1a) to calculate $L^1(D)$ and $L^\infty(D)$ errors. Plot $L^1(D)$ and $L^\infty(D)$ errors vs numbers of cells (for no. of cells 100, 200, 400, 800, 1600), for each case of the initial conditions in (6.1.2) - (6.1.4).

HINT: The runs of the FVM solver for meshes with 800 and 1600 cells can take several minutes.

Problem 6.2 Shallow water equations in two dimensions

Consider the 2-dimensional shallow water equations

$$\begin{cases} h_t + (hu)_x + (hv)_y = 0, \\ (hu)_t + \left(hu^2 + \frac{1}{2}gh^2\right)_x + (huv)_y = -ghb_x, \\ (hv)_t + (huv)_x + \left(hv^2 + \frac{1}{2}gh^2\right)_y = -ghb_y. \end{cases} \tag{6.2.1}$$

Here, for a point $(x, y) \in \mathbb{R}^2$ and time instance t, variable $h(x, y, t)$ denotes the height of the fluid column above the bottom topography $b = b(x, y)$ over which the fluid flows and $(u, v) = (u(x, y, t), v(x, y, t))$ is the vertically averaged (or depth averaged) horizontal fluid velocity field. The constant g denotes the size of the negative vertical acceleration due to gravity and is set to $g = 9.812$ here. For now, we also consider flat bottom topography, i.e. we set $b \equiv \text{const} = 0$, and hence the right hand side of (6.2.1) becomes zero.

Denoting the vectors of conserved variables (h, hu and hv) as $\mathbf{U} = \mathbf{U}(\mathbf{x}, t) : \mathbb{R}^2 \times \mathbb{R}_+ \to \mathbb{R}^3$, and the directional (in x and y directions) fluxes as $\mathbf{F}, \mathbf{G} : \mathbb{R}^3 \to \mathbb{R}^3$, i.e.

$$\mathbf{U} = \begin{bmatrix} h \\ hu \\ hv \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \\ huv \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} hv \\ huv \\ hv^2 + \frac{1}{2}gh^2 \end{bmatrix}, \tag{6.2.2}$$

the system (6.2.1) with given initial data $\mathbf{U}_0$ is rewritten as a *system of conservation laws*,

$$\begin{cases} \mathbf{U}(\mathbf{x}, t)_t + \mathbf{F}(\mathbf{U})_x + \mathbf{G}(\mathbf{U})_y = 0, \\ \mathbf{U}(\mathbf{x}, 0) = \mathbf{U}_0(\mathbf{x}). \end{cases} \quad \mathbf{x} = (x, y) \in \mathbf{D}, \quad t > 0, \tag{6.2.3}$$

where we restrict the computational domain to some bounded Cartesian domain $\mathbf{D} \subset \mathbb{R}^2$.

The maximum directional wave speeds (in directions x and y), corresponding to the maximal eigenvalues of the matrices $\partial\mathbf{F}/\partial\mathbf{U}$ and $\partial\mathbf{G}/\partial\mathbf{U}$, are given by the

$$\lambda_x(\mathbf{U}) = |u| + \sqrt{gh}, \quad \lambda_y(\mathbf{U}) = |v| + \sqrt{gh}. \tag{6.2.4}$$

(6.2a) As a starting point, we first consider the one-dimensional version of the shallow water equations, obtained by setting $v \equiv 0$ in (6.2.1), removing flux $\mathbf{G}$, and denoting $\mathbf{U} = (h, hu)^\top$,

$$\begin{cases} h_t + (hu)_x = 0, \\ (hu)_t + \left(hu^2 + \frac{1}{2}gh^2\right)_x = 0. \end{cases} \tag{6.2.5}$$

For the finite domain $\mathbf{D} = I_1 = (0, 2)$, implement the Finite Volume solver using the Rusanov numerical flux and the Forward Euler time stepping, i.e.

$$\mathbf{U}_i^{n+1} := \mathbf{U}_i^n - \frac{\Delta t}{\Delta x}(\mathbf{F}_{i+\frac{1}{2}}^n - \mathbf{F}_{i-\frac{1}{2}}^n), \tag{6.2.6}$$

where $\mathbf{U}_i^n$ are the vectors $(h_i^n, (hu)_i^n)^\top$ denoting the cell averages of the solution $\mathbf{U} = (h, hu)^\top$, and $\mathbf{F}_{i+\frac{1}{2}}^n$ are approximated by the Rusanov flux function

$$\mathbf{F}_{i+\frac{1}{2}}^n(\mathbf{U}_i^n, \mathbf{U}_{i+1}^n) \approx \mathbf{F}^{\text{Rus}}(\mathbf{U}_L, \mathbf{U}_R) = \frac{(\mathbf{F}_L + \mathbf{F}_R)}{2} - \frac{\lambda_{\max}}{2}(\mathbf{U}_R - \mathbf{U}_L), \tag{6.2.7}$$

with *local* maximum wave speeds

$$\lambda_{\max} := \max(\lambda_x(\mathbf{U}_L), \lambda_x(\mathbf{U}_R)).$$

The mesh width is $\Delta x = |I_1|/N_x$, where N_x denotes the number of the mesh cells, and the time step size respects the CFL condition (with CFL number C_{CFL} set to 0.9)

$$\Delta t = C_{\text{CFL}} \frac{\Delta x}{\bar{\lambda}_x} \leq \frac{\Delta x}{\bar{\lambda}_x} = \frac{\Delta x}{\max_i \lambda_x(\mathbf{U}_i)}. \tag{6.2.8}$$

Use the "outflow" boundary conditions, which can be easily implemented by extrapolating the values of the variable to the ghost cell i.e. $\mathbf{U}_0^n = \mathbf{U}_1^n$ and $\mathbf{U}_{N+1}^n = \mathbf{U}_N^n$.

Run your code for the "dambreak" initial data given by

$$\mathbf{U}_0(x) = \Big(h_0(x), u_0(x)\Big)^\top = \begin{cases} (2 - b(x), 0)^\top, & \text{if} \quad x < 1, \\ (1.5 - b(x), 0)^\top, & \text{otherwise}, \end{cases} \tag{6.2.9}$$

for $N_x = 512$ mesh cells up to final time $T = 0.1$ and plot the results (both water column height h and velocity u).

HINT: Consider the evaluations of $\mathbf{U}$ at the cell mid-points x_i for the cell averages $\mathbf{U}_i$.

HINT: Implement the Rusanov flux (6.2.7) as a separate function; this way you will be able to quickly extend and adapt it for the 2-dimensional FVM solver in the next sub-problem.

HINT: The solution consists of the left-moving rarefaction wave and the right-moving shock wave, as depicted in Figure 6.1.

HINT: It is advisable to implement and use the conversion functions between the *observable* (also called *primitive*) variables (h, u) and the *conserved* variables (h, hu).

HINT: You might find the MATLAB function `diff` or the Python function `numpy.diff` useful.

HINT: For debugging, it is always a good idea to plot the initial data and the solution after *one* time step, and verify that your numerical flux is correct and provides stable approximations.

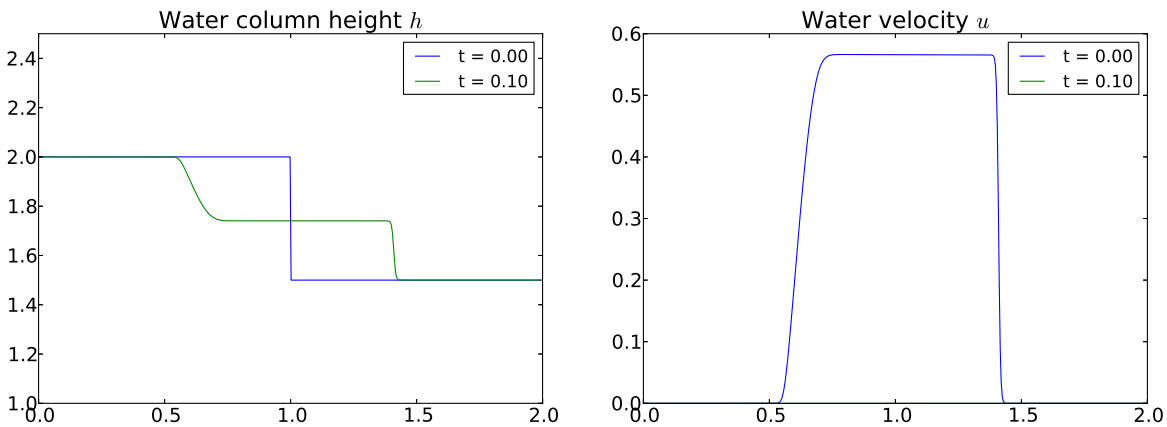

Figure 6.1: FVM approximations of the 1-D shallow water equations (6.2.5) and the dambreak problem (6.2.9) with flat bottom topography $b \equiv 0$.

(6.2b) Extend your code of subproblem (6.2a) to non-constant bottom topography, given by

$$b(x) = 0.1\sin(5\pi x) - 0.2x + 1.4. \tag{6.2.10}$$

The resulting FVM scheme then also needs to incorporate the source term $\mathbf{S}_i$,

$$\mathbf{U}_i^{n+1} := \mathbf{U}_i^n - \frac{\Delta t}{\Delta x}(\mathbf{F}_{i+\frac{1}{2}}^n - \mathbf{F}_{i-\frac{1}{2}}^n) + \Delta t \mathbf{S}_i, \tag{6.2.11}$$

which you can approximate by using second order accurate central differences,

$$\mathbf{S}_i = \begin{bmatrix} 0 \\ gh_i \frac{B_{i+1} - B_{i-1}}{2\Delta x} \end{bmatrix}. \tag{6.2.12}$$

I order to deal with the boundary conditions for b, simply set

$$\mathbf{S}_0 = (0,0)^\top, \quad \mathbf{S}_{N_x} = (0,0)^\top. \tag{6.2.13}$$

Run your code for the "dambreak" initial data (6.2.9) for $N_x = 512$ mesh cells up to final time $T = 0.1$ and plot the results (plot water surface $h + b$ and bottom topography b in the same plot).

HINT: Do not forget to *subtract* bottom topography $b(x)$ from $h(x)$ in (6.2.9).

HINT: For plotting, do not forget to *add* bottom topography b to water column h to get the actual water surface.

HINT: The solution, analogously as in subproblem (6.2a), consists of the left-moving rarefaction wave and the right-moving shock wave, as depicted in Figure 6.2.

(6.2c) In this sub-problem, we extend the 1-dimensional Finite Volume scheme from the subproblem (6.2a) with flat bottom topography $b \equiv 0$ to the 2-dimensional shallow water equations (6.2.3). Consider the finite domain $\mathbf{D} = I_1 \times I_2 = (0,2) \times (0,1)$ and the corresponding uniform axiparallel equidistant mesh with N_x cells in x direction and N_y cells in y direction. Implement the Finite Volume solver using the dimension splitting, Rusanov numerical flux and the Forward Euler time stepping, i.e.

$$\mathbf{U}_{i,j}^{n+1} := \mathbf{U}_{i,j}^n - \frac{\Delta t}{\Delta x}(\mathbf{F}_{i+\frac{1}{2},j}^n - \mathbf{F}_{i-\frac{1}{2},j}^n) - \frac{\Delta t}{\Delta y}(\mathbf{G}_{i,j+\frac{1}{2}}^n - \mathbf{G}_{i,j-\frac{1}{2}}^n), \tag{6.2.14}$$

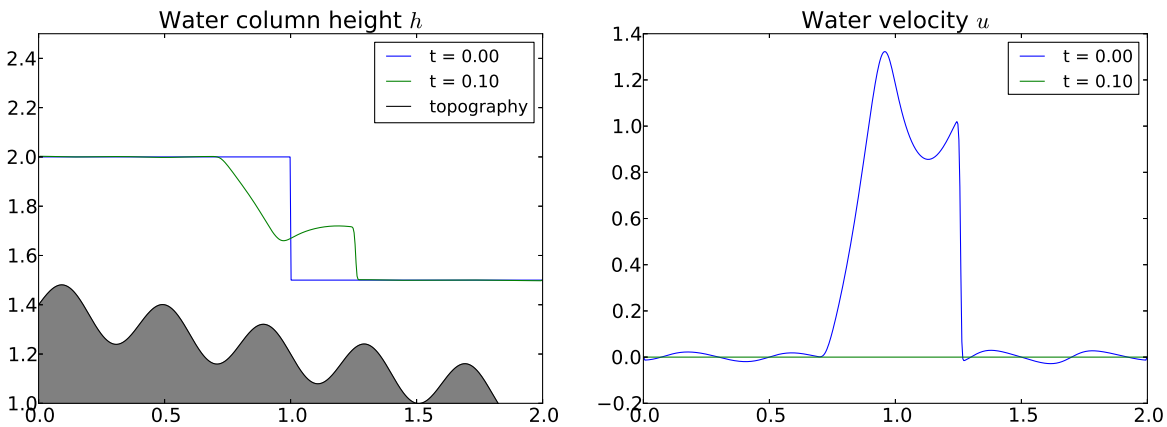

Figure 6.2: FVM approximations of the 1-D shallow water equations (6.2.5) and the dambreak problem (6.2.9) with varying bottom topography (6.2.10).

where $\mathbf{U}^n_{i,j}$ are the vectors $(h^n_{i,j}, (hu)^n_{i,j}, (hv)^n_{i,j})^\top$ denoting the cell averages of the solution $\mathbf{U} = (h, hu, hv)^\top$, and the *directional* fluxes $\mathbf{F}^n_{i+\frac{1}{2},j}$ and $\mathbf{G}^n_{i,j+\frac{1}{2}}$ are approximated by the Rusanov flux function from (6.2.7) in the *corresponding direction*, i.e.

$$\mathbf{F}^n_{i+\frac{1}{2},j} \approx \mathbf{F}^{\text{Rus}}(\mathbf{F}^n_{i,j}, \mathbf{F}^n_{i+1,j}), \quad \mathbf{G}^n_{i,j+\frac{1}{2}} \approx \mathbf{F}^{\text{Rus}}(\mathbf{F}^n_{i,j}, \mathbf{F}^n_{i,j+1}). \tag{6.2.15}$$

The time step size respects the 2-dimensional version of the CFL condition (6.2.8)

$$\Delta t = C_{\text{CFL}} \left(\frac{\bar{\lambda}_x}{\Delta x} + \frac{\bar{\lambda}_y}{\Delta y} \right)^{-1}, \quad \bar{\lambda}_x = \max_{i,j} \lambda_x(\mathbf{U}_{i,j}), \quad \bar{\lambda}_y = \max_{i,j} \lambda_y(\mathbf{U}_{i,j}). \tag{6.2.16}$$

Run your code for the 2-dimensional "dambreak" initial data given by

$$\mathbf{U}_0(x,y) = \Big(h_0(x,y), u_0(x,y), v_0(x,y)\Big)^\top = \begin{cases} (2 - b(x,y), 0, 0)^\top, & \text{if } x < (y-0.5)^2 + 0.75, \\ (1.5 - b(x,y), 0, 0)^\top, & \text{otherwise}, \end{cases} \tag{6.2.17}$$

for $N_x = 64$ and $N_y = 32$ mesh cells up to final time $T = 0.1$ and plot the results (both water column height h and velocities u and v). You might also want (this is optional) to generate a 3D plot with the water surface h being plotted on the z-axis. Use the "outflow" boundary conditions, which can be easily implemented by extrapolating the values of the variable to the ghost cell, i.e.

$$\mathbf{U}^n_{0,j} = \mathbf{U}^n_{1,j}, \quad \mathbf{U}^n_{N+1,j} = \mathbf{U}^n_{N,j} \tag{6.2.18}$$

in the x direction (i.e. for the evaluations of the flux $\mathbf{F}$), and

$$\mathbf{U}^n_{i,0} = \mathbf{U}^n_{i,1}, \quad \mathbf{U}^n_{i,N+1} = \mathbf{U}^n_{i,N}. \tag{6.2.19}$$

in the y direction (i.e. for the evaluations of the flux $\mathbf{G}$).

HINT: You might find the MATLAB function `meshgrid` or the Python function `meshgrid` (with option `indexing='ij'`) from `numpy` useful.

HINT: If you are eager to test the limits of your machine and prepared to wait longer (an hour or so), run your 2-D code with $N_x = 512$ and $N_y = 256$. For even larger resolutions (and hence smaller errors), parallel implementations are usually needed.

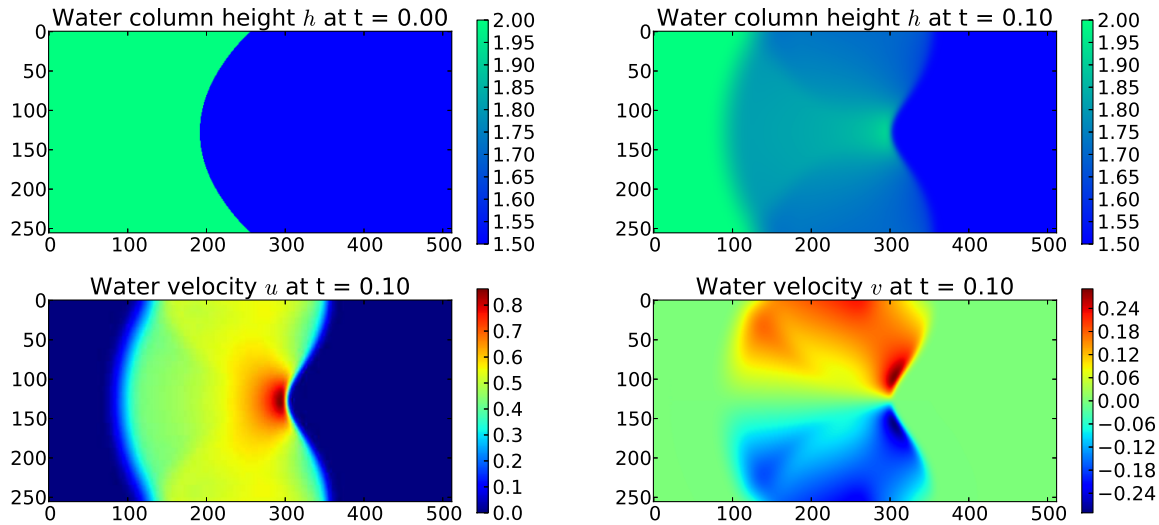

Figure 6.3: FVM approximations of the 2-D shallow water equations (6.2.1) and the dambreak problem (6.2.17).

HINT: For code testing purposes, the reference plots (using $N_x = 512$ and $N_y = 256$) are given in Figure 6.3.

Published on May 14th.

To be submitted on May 30th.

Last modified on May 16, 2014