ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

SCHOOL OF BASIC SCIENCES

MATH-459

NUMERICAL METHODS FOR CONSERVATION LAWS

Project 1

Solving a Non-Linear System of Conservation Laws

Lecturer: Martin Werner Licht
Teaching Assistant: Fernando José Henriquez Barraza

March 29, 2022

Lorenz Veithen 346364 Mathieu Grondin 296769

Contents

Introduction 1

This project investigates numerical solutions to the Shallow Water problem using both the Lax-Friedrichs and the Roe solver Finite Volume Schemes. The questions of the project will be answered in the related chapters. The one dimensional shallow water equation is given by Equation 1.1.

$$\begin{bmatrix} h \\ m \end{bmatrix}_t + \begin{bmatrix} m \\ \frac{m^2}{h} + \frac{1}{2}gh^2 \end{bmatrix}_x = \mathbf{Sc}(x,t), \tag{1.1}$$

where $h = h(x,t)$ is the *depth* or the height of the water and $m = m(x,t)$ is a quantity usually called the *discharge*, which measures the flow rate of the fluid past a point. g is the acceleration due to gravity (which is taken $g = 1\,m/s^{-2}$ in this work) and **Sc** $= \mathbf{Sc}(x,t)$ is a source term. It is further noted that Equation 1.1 has the form $\mathbf{q}_t + \mathbf{f}(\mathbf{q})_x = \mathbf{S}(x,t)$ with $\mathbf{q} = [h\ m]^T$. Hence, the horizontal velocity is defined as $u = m(x,t)/h(x,t)$.

Lax-Friedrichs Solver 2

In this chapter, the implementation of the Lax-Friedrichs (LF) method, to solve the non-linear system of conservation laws, is presented. The numerical scheme is then tested against using an initial value problem of which the exact solution is known. Finally, the scheme was used to compute the solution of two other initial value problems with unknown exact solution.

2.1 The Lax-Friedrichs Method

In this section, the implementation of the LF method for the shallow water problem will be presented. The domain $\Omega = [0, 2]$ is discretized into N cells of size dx of which the mid-point x_c is considered for the rest of the scheme. All following numerical expressions make use of cell average values centred on those midpoints. The general system of conservation laws from Equation 2.1 is considered.

$$\begin{bmatrix} h \\ m \end{bmatrix}_t + \begin{bmatrix} m \\ \frac{m^2}{h} + \frac{1}{2}gh^2 \end{bmatrix}_x = \mathbf{Sc}(x,t), \tag{2.1}$$

Which can be rewritten in the form given by Equations 2.2 and 2.3.

$$\begin{bmatrix} h \\ m \end{bmatrix}_t + A(m,h)\begin{bmatrix} h \\ m \end{bmatrix}_x = \mathbf{Sc}(x,t), \tag{2.2}$$

$$A(m,h) = \begin{bmatrix} 0 & 1 \\ -(\frac{m^2}{h^2} - gh) & \frac{2m}{h} \end{bmatrix} \tag{2.3}$$

Which is of the form $\mathbf{q}_t + A(m,h)\mathbf{q}_x = \mathbf{Sc}(x,t)$ with $\mathbf{q} = [h\ m]^T$. And where $A = A(m,h)$ clearly shows the non-linearity of the problem and A will change depending on the values of $h = h(x,t)$ and $m = m(x,t)$ at the cell and time considered. This is taken care of by using the values of h and m at that position from the previous time step, relying on the assumption that those variables do not change too much on one time step. The implementation therefore considers two nested loops: one in time and one in space, as shown in Figure 2.1.

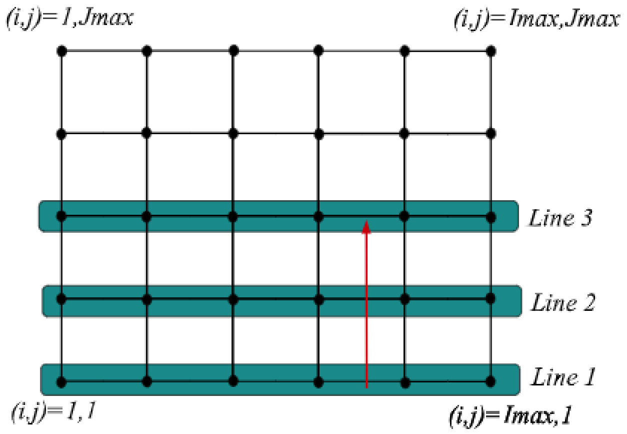

Figure 2.1: Space and time marches. j in the time direction and i in the x direction.

First, all the x_c positions are iterated upon such that a local A matrix can be created using the values of h and m from the previous time step at the considered position. Considering the local matrix A_i at the i^{th} position x_c, the problem can be considered as locally linear and solved using linear methods. Therefore, the local equation

$\mathbf{q}_t + A_i\mathbf{q}_x = \mathbf{Sc}(x,t)$ is solved by decoupling the system using eigenvalue decomposition, assuming that A_i is diagonalisable. Determining the eigenvalues λ_i and eigenvectors $\mathbf{s}_i$ of A_i,

$$\Lambda_i = \begin{bmatrix}\lambda_1 & 0\\ 0 & \lambda_2\end{bmatrix} \qquad S_i = [\mathbf{s}_1\ \mathbf{s}_2] \qquad \Lambda_i = S_i^{-1}A_iS_i$$

By left multiplying by S_i^{-1} and making $I = S_iS_i^{-1}$ appear, the local equation can be written as Equation 2.4.

$$\mathbf{V}_t + \Lambda_i\mathbf{V}_x = S_i^{-1}\mathbf{Sc}(x,t) \tag{2.4}$$

With $\mathbf{V} = S_i^{-1}\mathbf{q} = [v_1\ v_2]^T$. This system is decoupled and can be solved as two distinct equations of $v_1(x,t)$ and $v_2(x,t)$, of the form $v_t+\lambda v_x = Sc(x,t)$ which is a transportation equation which can be solved numerically using the Lax-Friedrichs scheme. Using,

$$k = CFL\frac{dx}{max_i(|m_i/h_i| + \sqrt{(gh_i)}}$$

where the values of m and h are taken from the previous time step and the CFL condition is taken as 0.5. The LF method can be constructed the LF flux given by Equation 2.5 and the conservative form of numerical methods given by Equation 2.6.

$$F(u,v) = \frac{1}{2}\left(f(u) + f(v) - \frac{dx}{k}(v-u)\right) \tag{2.5}$$

$$v_i^{n+1} = v_i^n - \frac{k}{dx}(F^n_{i+\frac{1}{2}} - F^n_{i-\frac{1}{2}}) + k\cdot Sc \tag{2.6}$$

Where $f(u) = \lambda u$, n is the previous time step, i denotes the position of the central node of one cell and Sc is the relevant component of $S_i^{-1}\mathbf{Sc}(x,t)$. The explicit scheme to update one component of **V**, therefore writes (combining Equations 2.5 and 2.6):

$$v_i^{n+1} = \frac{1}{2}(v^n_{i+1} - v^n_{i-1}) - \frac{\lambda k}{2dx}(v^n_{i+1} - v^n_{i-1}) + k\cdot Sc$$

Where λ is λ_1 or λ_2 depending on which component of **V** is being solved. The last step in this process is then to obtain the vector **q** from $\mathbf{q} = S_i\mathbf{V}$, which yields the values of h_i and m_i at the current time step. Those values are stored until the loop in x_c is completely finished, then the new values are used for the next iteration.

2.2 Testing the Code

As asked in question 1.1b, the code is tested on the initial value problem described by Equations 3.6 and 3.7, and using the source term described by Equation 3.8 (with $u = 0.25$).

$$h(x,0) = h_0(x) = 1 + 0.5\sin(x) \tag{2.7}$$

$$m(x,0) = m_0(x) = uh_0(x) \tag{2.8}$$

$$\mathbf{Sc} = \begin{bmatrix}\frac{\pi}{2}(u-1)\cos\big(\pi(x-t)\big)\\ \frac{\pi}{2}\cos\big(\pi(x-t)\big)(-u+u^2+gh_0(x-t))\end{bmatrix} \tag{2.9}$$

As the solver described earlier is based on the Finite Volume Method, the initial values and the source need to be discretized into cell averages. This is done by taking the integral on each cell and dividing by the cell length, this is shown in Equations 2.10 and 2.11 which gives the value of the initial value and the components of the source term on the ith cell.

$$\frac{1}{dx}\int_{x_i}^{x_{i+1}} h_0(y)\,dy = 1 - \frac{0.5}{\pi dx}(cos(\pi x_{i+1}) - \cos(\pi x_i)) \tag{2.10}$$

$$\frac{1}{dx}\int_{xi}^{x_{i+1}} S_1(y,t)dy = (u-1)/(2dx))(\sin(\pi * (x_{i+1}-t)) - \sin(\pi(x_i - t)) \tag{2.11}$$

$$\begin{aligned}\frac{1}{dx}\int_{xi}^{x_{i+1}} S_2(y,t)dy =& \frac{1}{dx}(\frac{1}{2}(g+(u-1)u)(\sin(\pi(x_{i+1}-t)) - \sin(\pi(x_i-t))) \\ &- 0.0625g(\cos(2\pi(x_{i+1}-t)) - \cos(2\pi(x_i-t))))\end{aligned}$$

Note that x takes the values of the positions at the interfaces between the cells. This was implemented with periodic boundary conditions, yielding the results showed on Figure 2.2 for $m(x,t)$ and $h(x,t)$ at $T = 2s$.

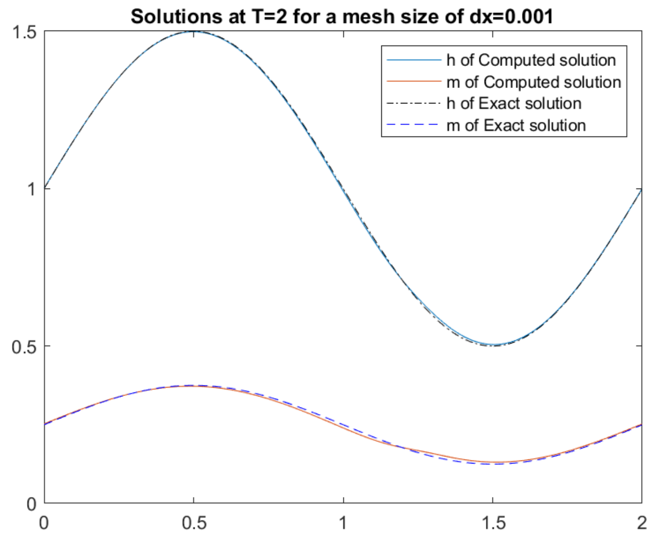

Figure 2.2: Numerical and exact solution of exercise 1.1.b at $T = 2s$ with $dx = 0.001$. The horizontal axis represents the spacial dimension while the vertical axis gives the value of $h(x,2)$ and $m(x,2)$

Clearly, the solution obtained numerically using the method outlined in the previous section reaches the entropy condition and therefore approaches correctly the exact solution given by $h(x,t) = h_0(x-t)$ and $m(x,t) = u \cdot h(x,t)$ with u=0.25. It can be observed that some numerical dissipation takes place, as can be expected from the flux used. Furthermore, the scheme can be shown to be of order one by analysing the error on a log-log plot with the mesh size, as shown in Figure 2.3. This was done using the infinite norm $\|.\|_\infty$ for the error.

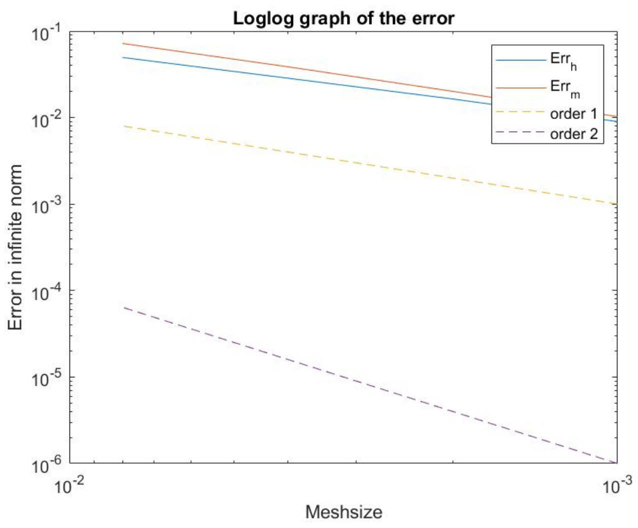

Figure 2.3: Error of the numerical solution of exercise 1.1b at $T = 2s$ with mesh sizes of 0.008, 0.004, 0.002, 0.001

2.3 Using the Scheme on Problems of Unknown Exact Solution

Now that it has been verified that the method converges using a small enough mesh size, the same program is applied to problems of unknown exact solution with different initial conditions and no source term. For each, this is done using periodic boundary conditions and the solution at $T = 2\,s$ with several mesh sizes is compared to a very fine solution in order to assess the order of convergence of the scheme.

2.3.1 Question 1.2.a

The scheme is applied to the following initial condition with a 0 source term:

$$h_0(x) = 1 - 0.1\sin(\pi x) \qquad m_0(x) = 0 \tag{2.12}$$

The discretization into cell averages is simply given by the following:

$$\frac{1}{dx}\int_{x_i}^{x_{i+1}} h_0(y)dy = 1 + \frac{0.1}{\pi dx}(\cos(\pi x_{i+1}) - \cos(\pi x_i)) \tag{2.13}$$

Again, the x values refer to the boundaries of each cell. The solution is given by Figure 2.4 and the log-log plot of the error is shown in Figure 2.5.

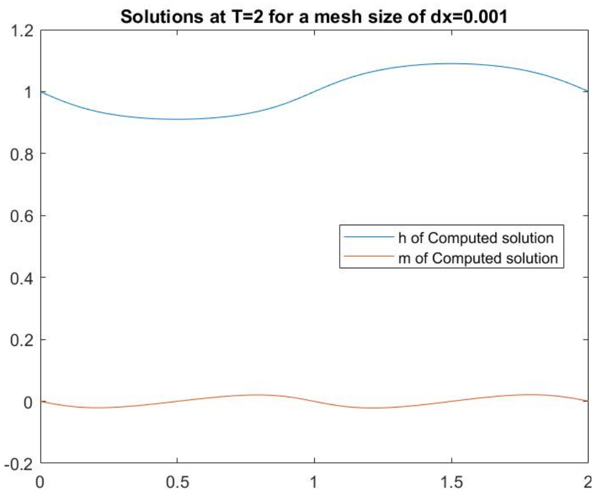

Figure 2.4: Numerical solution of exercise 1.2.a at $T = 2s$ with $dx = 0.001$. The horizontal axis represents the spacial dimension while the vertical axis gives the value of $h(x, 2)$ and $m(x, 2)$

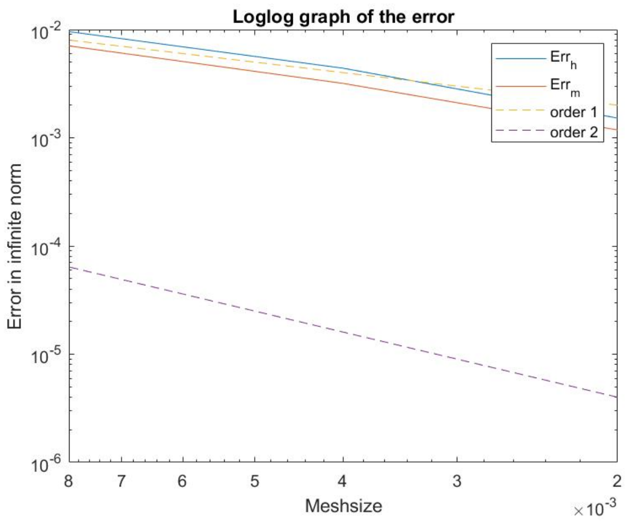

Figure 2.5: Error of the numerical solution of exercise 1.1b at $T = 2s$ with mesh sizes of 0.008, 0.004, 0.002, 0.001

For this problem, it can again be seen that the order of convergence is close to 1. The same L_∞ norm was used, and the reference mesh size was taken as $dx = 0.001$.

2.3.2 Question 1.2.b

This is again repeated with the following initial value problem:

$$h_0(x) = 1 - 0.2\sin(2\pi x) \qquad m_0(x) = 0.5 \tag{2.14}$$

The initial value is again discretized in the following manner,

$$\frac{1}{dx}\int_{x_i}^{x_{i+1}} h_0(y)dy = 1 + \frac{0.2}{2\pi dx}(\cos(\pi x_{i+1}) - \cos(\pi x_i)) \tag{2.15}$$

This then yields the solution shown in Figure 2.6 with a fine mesh of $dx = 0.001$ (2000 cells). The error behaviour was plotted in Figure 2.7.

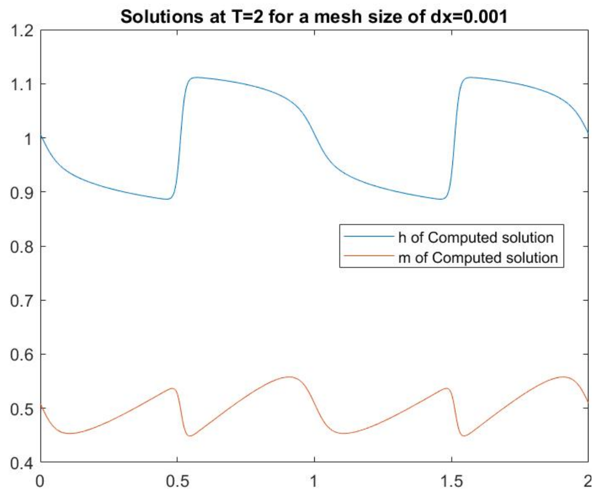

Figure 2.6: Numerical solution of exercise 1.2.b at $T = 2s$ with $dx = 0.001$. The horizontal axis represents the spacial dimension while the vertical axis gives the value of $h(x,2)$ and $m(x,2)$

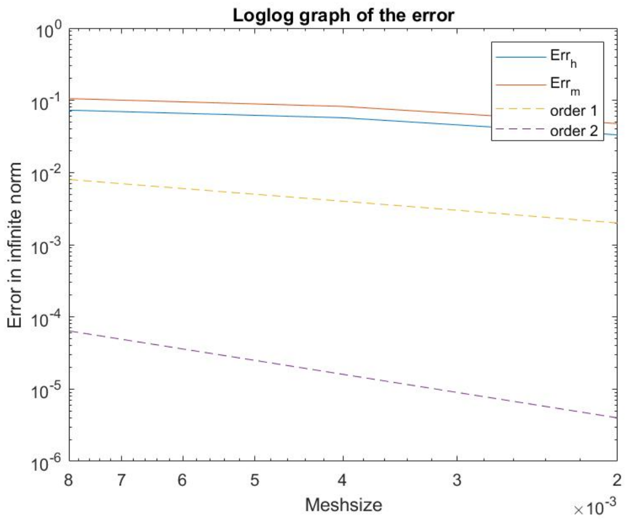

Figure 2.7: Error of the numerical solution of exercise1.1b at $T = 2s$ with mesh sizes of 0.008, 0.004, 0.002,0.001

2.4 Discontinuous Initial Value

In this section, questions 1.4.a and 1.4.b are tackled. The last part of the question is detailed in the part dedicated to the Roe Solver of this document. In this case, a discontinuous initial value in the discharge of the shallow water is introduced. The initial value problem given by Equation 2.16 is approximated using the Lax-Friedrichs Solver.

$$h_0(x) = 1 \qquad m_0(x) = \begin{cases} -1.5 & x < 1 \\ 0 & x > 1 \end{cases} \tag{2.16}$$

This is again discretised in the same manner as before, although $h_0(x)$ does not require any care in this case and $m_0(x)$ requires care on the cell containing the value $x = 1$. It was rapidly noticed that the solver described in section 2.1 induces oscillations in the approximated solution. This is the case due to the eigenvalue decomposition that is used. To cope with this issue, an alternative method which does not require any eigenvalue decomposition was implemented (and was found to give the same accuracy as the previously detailed scheme while being more efficient in terms of the number of required operations).

In this method, no matrix $A(m,h)$ is derived and the formulation of the flux (see Equation 2.17) as given in the project description is used.

$$f(m,h) = \begin{bmatrix} m \\ \frac{m^2}{h} + \frac{1}{2}gh^2 \end{bmatrix} \tag{2.17}$$

The Lax-Friedrichs numerical flux is then used in the conservative form of numerical schemes. This is shown in Equation 2.18.

$$\mathbf{q}_i^{n+1} = \mathbf{q}_i^n - \frac{k}{dx}\left(F^{LF}(\mathbf{q}_i, \mathbf{q}_{i+1}) - F^{LF}(\mathbf{q}_{i-1}, \mathbf{q}_i)\right) \qquad F^{LF}(u,v) = \frac{1}{2}\left(f(u) + f(v) - \frac{dx}{k}(v-u)\right) \tag{2.18}$$

In this manner, both $h_i(x,t) = \mathbf{q}_i(1,1)$ and $m_i(x,t) = \mathbf{q}_i(2,1)$ are updated in one operation. A reference solution on a mesh using $dx = 0.0001$ (20000 points) is obtained by applying this method to the discontinuous initial value problem using open boundary conditions. Note that this mesh is finer than for the previous problems, as the discontinuity is more difficult to model and therefore requires more care. The reference solution is shown in Figure 2.8. Furthermore, the experiment is repeated for multiple mesh sizes and the solutions are compared to this reference solution. This is given by Figures 2.9 and 2.10.

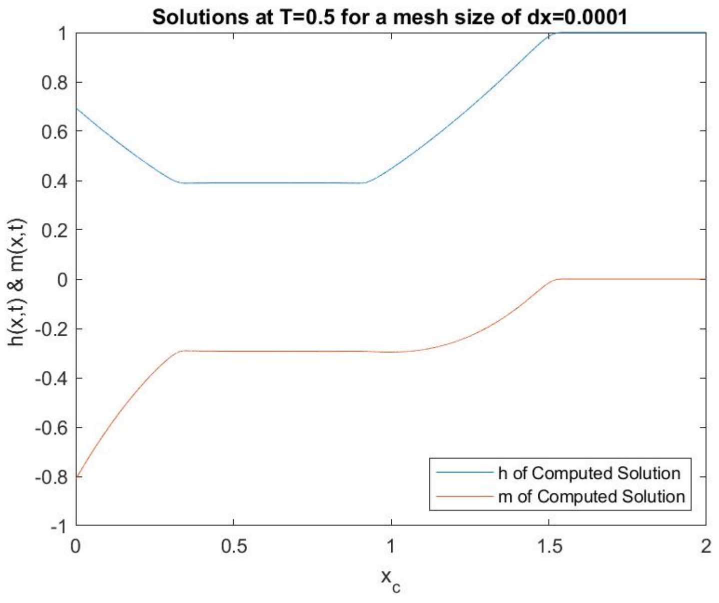

Figure 2.8: Reference solution to question 1.4 at $T = 0.5s$ with a mesh size $dx = 0.0001$

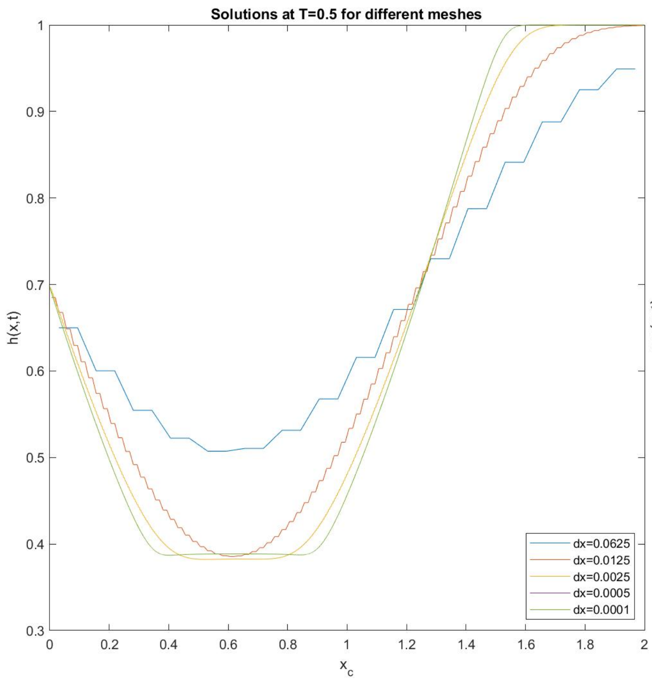

Figure 2.9: Numerical solutions of problem 1.4.b $h(x,t)$ for different mesh sizes and compared to the reference solution obtained earlier

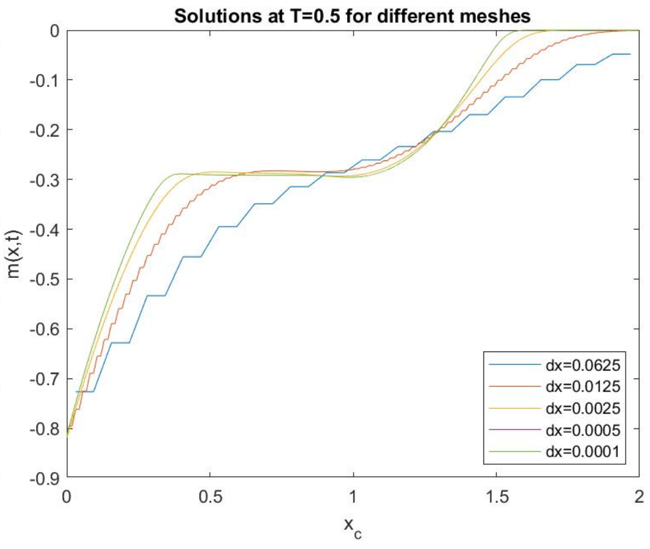

Figure 2.10: Numerical solutions of problem 1.4.b $m(x,t)$ for different mesh sizes and compared to the reference solution obtained earlier

It is clearly seen from Figures 2.9 and 2.10 that the Lax-Friedrichs scheme predicts two rarefaction waves in both the numerical solution of $m(x,t)$ and $h(x,t)$. As will be seen in the following chapter, this differs from the Roe Solver solution to the problem, which keeps a fidelity to the discontinuity. The rarefaction waves on both the $h(x,t)$ and $m(x,t)$ solutions are predicted on the intervals 0 to 0.3 and 1 to 1.5.

Roe Solver 3

In this chapter, Roe solver is implemented to solve the same problems as investigated in the Lax-Friedrichs chapter. First, a reasoning on why this problem pertains to the application of Godunov's method is given. This is followed by the derivation of the Roe matrix and an explanation of the algorithm that was implemented. Finally, the differences in the numerical methods obtained from both solvers is given.

3.1 Validity of the Method

This problem can be solved using Godunov's method as it consists of discontinuous interfaces between the cells on the spatial domain, assuming that Finite Volume Method is used. Figure 3.1 shows the setting at each cell interface.

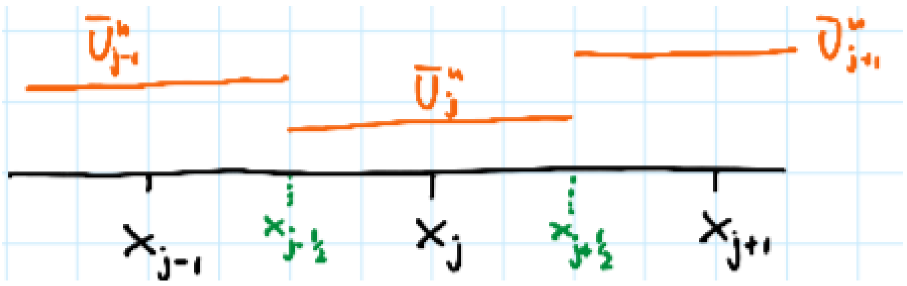

Figure 3.1: Drawing showing the Riemann problem at the ith cell interfaces. Taken from Lectures Notes 17 - Finite Volume Method

In Figure 3.1, the values are cell averages, as given by Equation 3.1. Though, note that the cell averages for the values $h(x,t)$ and $m(x,t)$ were also used in Lax-Fridriechs, only the numerical flux is different for this purpose.

$$\overline{U}_j^n = \frac{1}{n}\int_{x_{j-1/2}}^{x_{j+1/2}} u(x,t^n)dx \qquad \overline{F}_{j+1/2}^n = \frac{1}{k}\int_{t^n}^{t^{n+1}} f(u(x_{j+1/2},t))dt \tag{3.1}$$

$$\overline{U}_j^{n+1} = \overline{U}_j^n - \frac{k}{h}(\overline{F}_{j+1/2}^n - \overline{F}_{j-1/2}^n) \tag{3.2}$$

This formulation induces discontinuities at all cell interfaces in the approached solution, which can be considered as local Riemann problems to be solved exactly on a small enough time step. The size of such time step being dictated by the speed of the fastest wave in the system through CFL condition: $k < \frac{dx}{\lambda_{max}}$, where λ_{max} is the speed of such fastest wave. This condition ensures that no interaction between the different waves occurs.

Having solved those Riemann problems, $\overline{F}_{j+1/2}^n$ is given by considering the value $u(x,t)$, the solution, at the cell interfaces. One way to do so is to linearise the problem using a Roe matrix $\hat{A}$, as was seen in Lecture Notes 20 - Nonlinear systems, part 1. This matrix can then be used in the flux (where $|\hat{A}|$ will be defined in the following sections):

$$F(u,v) = \frac{1}{2}(f(u) + f(v)) - \frac{1}{2}|\hat{A}|(v-u)$$

3.2 Derivation of the Roe Matrix

The Godunov's method with a Roe linearization requires a matrix $\hat{A}$ such that : $\hat{A}(\mathbf{q}_r - \mathbf{q}_l) = \mathbf{f}_r - \mathbf{f}_l$. To do so, let us introduce a new parameter $z = (q_1)^{-1/2}\mathbf{q}$. Considering the notation of our problem $\left(\mathbf{q} = ([h\ m]^T\right)$, this is equivalent to $z = h^{-1/2}\mathbf{q}$. Determining the Roe matrix comes back to finding the matrices B and C such that $\mathbf{q}_r - \mathbf{q}_l = B(\mathbf{z}_r - \mathbf{z}_l)$ and $\mathbf{f}_r - \mathbf{f}_l = C(\mathbf{z}_r - \mathbf{z}_l)$. After finding these two matrices, and assuming that B is non-singular, one can recover $\hat{A}$ choosing $\hat{A} = CB^{-1}$. Indeed, it holds that : $CB^{-1}(\mathbf{q}_r - \mathbf{q}_l) = C(\mathbf{z}_r - \mathbf{z}_l) = \mathbf{f}_r - \mathbf{f}_l$.

First one can compute the matrix B by considering the following,

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} h^{1/2} \\ mh^{-1/2} \end{bmatrix} \Leftrightarrow \begin{bmatrix} z_1^2 \\ z_1 z_2 \end{bmatrix} = \begin{bmatrix} h \\ m \end{bmatrix}$$

$$\Delta h = \Delta z_1^2 = 2\overline{z_1}\Delta z_1 \qquad\qquad \Delta m = \Delta(z_1 z_2) = \overline{z_2}\Delta z_1 + \overline{z_1}\Delta z_2$$

where $\overline{\phi} = \frac{\phi_r + \phi_l}{2}$ and $\Delta\phi = \phi_r - \phi_l$ This corresponds to the following matrix B

$$B = \begin{bmatrix} 2\overline{z_1} & 0 \\ \overline{z_2} & \overline{z_1} \end{bmatrix}$$

With the same method the matrix C is obtained :

$$\Delta f_1 = \Delta m = \overline{z_2}\Delta z_1 + \overline{z_1}\Delta z_2$$

$$\Delta f_2 = \Delta\left(\frac{m^2}{h} + \frac{1}{2}gh^2\right)$$

$$\Leftrightarrow \Delta f_2 = \Delta(z_2^2) + \frac{1}{2}g\Delta(h^2)$$

$$\Leftrightarrow \Delta f_2 = 2\overline{z_2}\Delta z_2 + \frac{1}{2}g(2\overline{h}\Delta h)$$

$$\Leftrightarrow \Delta f_2 = 2\overline{z_2}\Delta z_2 + 2g\overline{z_1^2}\,\overline{z_1}\Delta z_1$$

Therefore, the matrix C can be written as:

$$C = \begin{bmatrix} \overline{z_2} & \overline{z_1} \\ 2g\overline{(z_1^2)}\,\overline{z_1} & 2\overline{z_2} \end{bmatrix}$$

Provided that $z_1 \neq 0$, B is non-singular so A can be computed as :

$$\hat{A} = CB^{-1} = \begin{bmatrix} \overline{z_2} & \overline{z_1} \\ 2g\overline{(z_1^2)}\,\overline{z_1} & 2\overline{z_2} \end{bmatrix} \begin{bmatrix} 2\overline{z_1} & 0 \\ \overline{z_2} & \overline{z_1} \end{bmatrix}^{-1}$$

$$\Leftrightarrow \hat{A} = \frac{1}{2\overline{z_1}^2} \begin{bmatrix} \overline{z_2} & \overline{z_1} \\ 2g\overline{(z_1^2)}\,\overline{z_1} & 2\overline{z_2} \end{bmatrix} \begin{bmatrix} \overline{z_1} & 0 \\ -\overline{z_2} & 2\overline{z_1} \end{bmatrix}$$

$$\Leftrightarrow \hat{A} = \frac{1}{2\overline{z_1}^2} \begin{bmatrix} 0 & 2\overline{z_1}^2 \\ 2\left(g(\overline{z_1})^2\overline{(z_1^2)} - (\overline{z_2})^2\right) & 4\overline{z_2}\,\overline{z_1} \end{bmatrix}$$

In conclusion the Roe matrix $\hat{A}$ is given by :

$$\hat{A} = \begin{bmatrix} 0 & 1 \\ \overline{(z_1^2)}g - (\overline{z_2})^2(\overline{z_1})^{-2} & 2\overline{z_2}(\overline{z_1})^{-1} \end{bmatrix} \tag{3.3}$$

It remains to verify that $\hat{A}$ is consistent with the flux Jacobian, ie. if we consider $z = z_l = z_r$ with $z = h^{-1/2}\mathbf{q}$ the Roe matrix is equal to the flux Jacobian. Note that,as $z_l = z_r = z$, $\overline{z_1} = z_1$ and the same holds for z_2 and even z_1^2. Hence :

$$\hat{A} = \begin{bmatrix} 0 & 1 \\ z_1^2 g - z_2^2 z_1^{-2} & 2z_2 z_1^{-1} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ hg - \frac{m^2}{h}\frac{1}{h} & 2\frac{m}{\sqrt{h}}\frac{1}{\sqrt{h}} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -(\frac{m^2}{h^2} - gh) & \frac{2m}{h} \end{bmatrix} = A(m,h)$$

So the consistency of the Roe matrix is verified.

3.3 Implementation of the Roe Solver

To implement the Godunov's method, the program uses the same discretization and the same initialisation as in the Lax-Friedriech's method. The method's differs in the executions between two time-steps.

Indeed in the Godunov's method, at each time-steps we have to compute the length of the time step following the CFL condition $\lambda_{max}\, k = h$ where λ_{max} is the maximum over all the absolute values of the eigenvalues over all the Roe matrices. This condition ensure that no interaction occurs between two neighbouring waves emerging from the Riemann problems.

In Godunov's method with the Roe solver, at each node the Roe matrix is different. So, at first sight, there is no easy condition on the time step. However, at each node x_c the Roe matrix is the same in term of z_l and z_r (the values of the variable defined in the previous section on the left and right cell to the one being considered). This is equivalent to say that $\hat{A}$ depends only on $\mathbf{q}_l$ and $\mathbf{q}_r$. Thus one can compute the eigenvalues as a function of the values at the right and left of the considered node. This is the reason why, in the algorithm, a function that receives two vectors and returns the λ_{max} is implemented. This function, named $eigvalA$, receives two vectors, the first one correspond to all the right values and the second for all the left values. Because the matrix $\hat{A}$ is expressed in z and not in $\mathbf{q}$, the transformation is first operated. Then all the eigenvalues are stored in a $2 \times n$ matrix.

Those eigenvalues are obtained thanks to the formulas:

$$\lambda_1 = \left(2\overline{z_2}\,\overline{z_1}^{-1} + \sqrt{4(\overline{z_2})^2(\overline{z_1})^{-2} + 4(\overline{(z_1^2)}g - (\overline{z_2})^2(\overline{z_1})^{-2})}\right)/2 \tag{3.4}$$

$$\lambda_2 = \left(2\overline{z_2}\,\overline{z_1}^{-1} - \sqrt{4(\overline{z_2})^2(\overline{z_1})^{-2} + 4(\overline{(z_1^2)}g - (\overline{z_2})^2(\overline{z_1})^{-2})}\right)/2 \tag{3.5}$$

Then the function just returns the overall maximum of the matrix. Because the condition is very important there is a scaling of 0.5, this ensures that even with round off errors the condition is respected.

When the time-step is fixed, the algorithm can solve one by one the local Riemann problems. To do so, there is a loop over all the mesh points, and for all points the special Roe matrix $\hat{A}$ is to be computed. This is done thanks to a function of two 2×1 vectors that returns the matrix. Indeed, it depends on the values at the left and the right of the segment. The first and the second vector correspond to the left and the right values respectively. Once again the inputs are expressed in terms of $\mathbf{q}$ so it first have to transform them into z-expressed vectors. After this, the matrix is computed using to the formula derived in the previous section.

At this moment of the program, one got nearly everything to compute the update. The update is implemented as a function of several parameters. The function should return the value of the vector $\mathbf{q}$ on the current segment for the current time-step. To do so, it needs the values at the previous, the current and the next nodes, respectively q_L, q_M and q_R , the time step previously computed, the Roe matrix $\hat{A}$, its eigenvalues in absolute values, stocked in a diagonal matrix, Λ, the size of the mesh dx and the value of the function $\mathbf{Sc}$ at the current node. The Roe flux is then implemented as follows, where $|\hat{A}| = S\hat{A}S^{-1}$:

$$F(u,v) = \frac{1}{2}(f(u) + f(v)) - |\hat{A}|(v-u)/2$$

Then the function just returns the value given by this formula :

$$q_{new} = q_M - \frac{k}{dx}(F(q_M, q_R) - F(q_L, q_M)) + k\mathbf{Sc_M}$$

When the new value is computed, it is stored in a vector of size $2 \times n$ named $\mathbf{q}_{new}$ and the loop restarts for the next node. Note that the values of the next nodes won't interact with the values already computed. Indeed the value of the current solution is replaced only when the loop is ended. The time is also updated and the program plot the actual solution. Finally, the loop can reiterate if the time is smaller than 2 second.

3.4 Applying the Roe Solver to Problems 1.1 and 1.2

For the question 1.1, the code is implemented with the given boundary conditions, in this case periodic, with the given initial values :

$$h(x,0) = h_0(x) = 1 + 0.5\sin(x) \quad (3.6) \qquad m(x,0) = m_0(x) = uh_0(x) \quad (3.7)$$

$$\mathbf{Sc} = \begin{bmatrix} \frac{\pi}{2}(u-1)\cos\big(\pi(x-t)\big) \\ \frac{\pi}{2}\cos\big(\pi(x-t)\big)(-u+u^2+gh_0(x-t)) \end{bmatrix} \tag{3.8}$$

where u=0.25. For the initial values, the code does not take the values in contiguous form, but as a vector of the size of the mesh with the average values of the initial conditions over the segments. The formulas were already computed for the previous chapters and are reused. The graph at time $T = 2$ is shown in figure 3.2 with the loglog graph of the error. For the error, the code is executed on different mesh sizes, and the error in infinite norm is stored in a vector. We can see the order of convergence of the error is 1.

For the question 1.2, the same method is used. The only things that change are the **Sc** function and the initial values. As above, the code do not use the initial values functions, but a vector. The formulas were already computed in the previous question, so we can just reuse the formulas. In the first case:

$$h_0(x) = 1 - 0.1\sin(\pi x) \qquad m_0(x) = 0 \tag{3.9}$$

The plot of the solution at time $T = 2$ and the loglog graph of the error are given in figure 3.3 The order of convergence of the error is 1. In the second case :

$$h_0(x) = 1 - 0.2\sin(2\pi x) \qquad m_0(x) = 0.5 \tag{3.10}$$

The plot of the solution at time $T = 2$ and the loglog graph of the error are given respectively in figure 3.4. The order of convergence in this particular case is 1 but the solution is not very satisfactory.

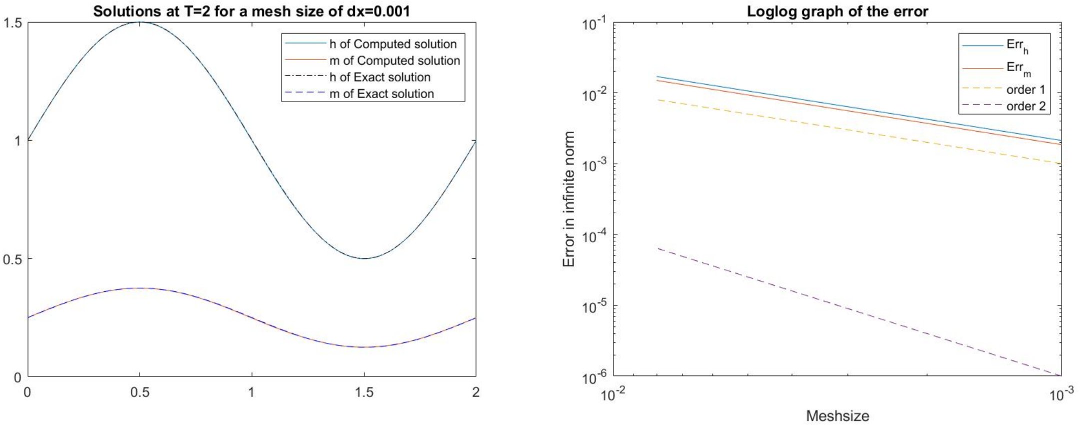

Figure 3.2: Question 1.1 with Godunov's method

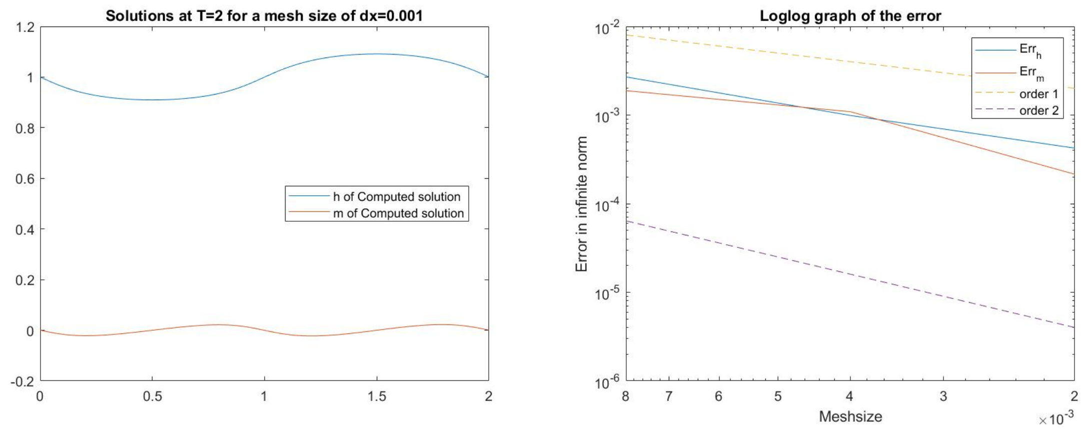

Figure 3.3: Question 1.2.a with Godunov's method

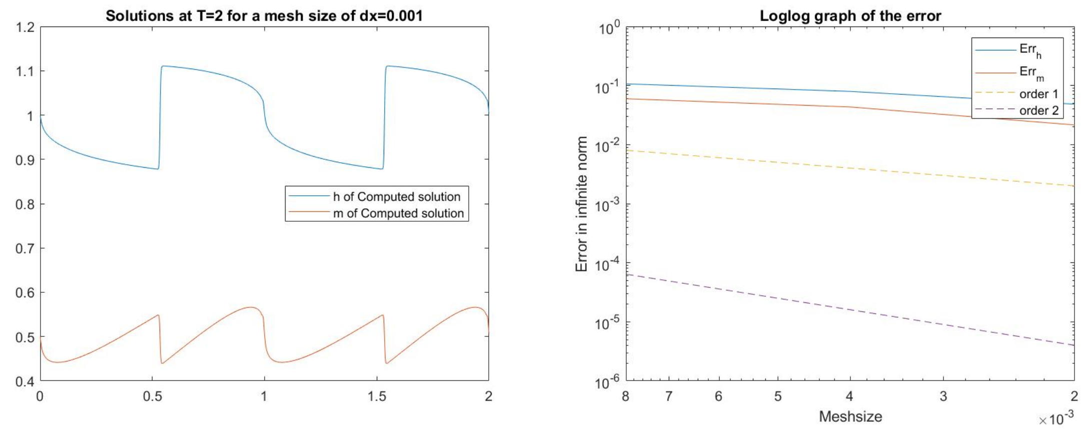

Figure 3.4: Question 1.2.b with Godunov's method

3.5 Accuracy of the Roe and LF Solvers

In terms of accuracy, both solvers seem to have the same order of convergence of one (as was seen earlier). However, the Lax-Friedrichs scheme tend to flatten and smooth the solution. This can be seen by comparing the solutions of problem 1.2.b using both the Roe and the LF solvers, as shown in Figure 3.5. Clearly, the Roe solver was more able to represent discontinuities in the form of shocks when considering the $m(x,t)$ numerical solution. On the contrary, the LF solver is flattened out more and shows a more wavy behaviour where a sharp behaviour is expected from the Roe solution. As will be seen in the next section, overall the Roe solver performs better when it comes to predicting the behaviour of discontinuities.

3.6 Solver Applied to Discontinuous Initial Value

The solutions of question 1.4.c with Roe solver are shown in Figures 3.6 and 3.7. Comparing to the reference solution computed with the Lax Friedrich method the solutions are consistently different. For both the $h(x,t)$ and $m(x,t)$ solutions, the rarefaction wave between $x = 0$ and $x = 0.45$ in the reference solution does not appear in the Roe solver solution. Instead, two visible shocks at $x = 0$ and $x = 0.45$ can be found. Indeed, as was previously noted, the Godunov's method keeps being faithful to discontinuities whereas Lax-Friedrich tends to smooth the solution. The only common wave is the last rarefaction wave between $x = 1$ and $x = 1.5$ on both the $h(x,t)$ and $m(x,t)$ solution, it can therefore be found that the Roe solver converges to the LF solution for $x \in [0.45, 2]$ considering $m(x,t)$ and on $x \in [1, 2]$ considering $h(x,t)$. Overall, the mesh size makes the Roe solution convergence in those regions but not on the whole domain.

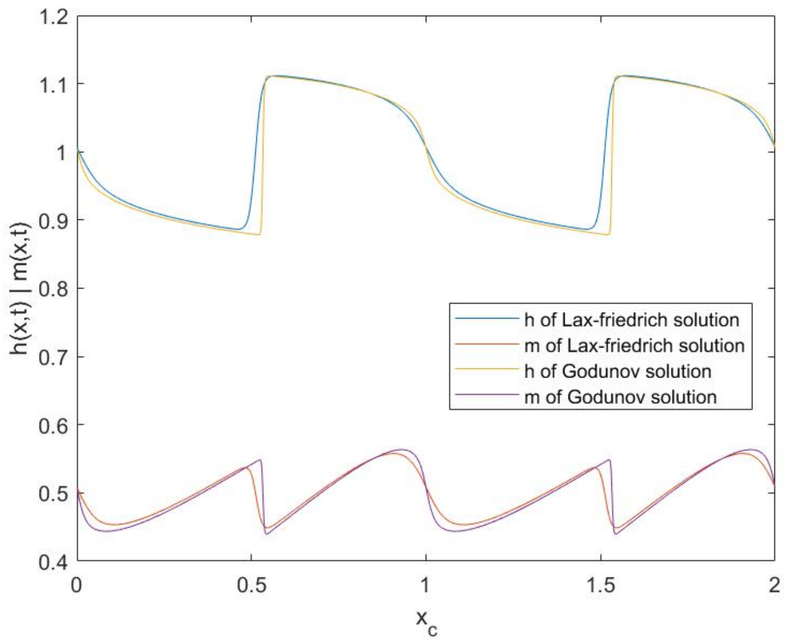

Figure 3.5: Comparison Roe and LF solver solutions of problem 1.2.b at $T = 2s$ on a mesh size of $dx = 0.001$

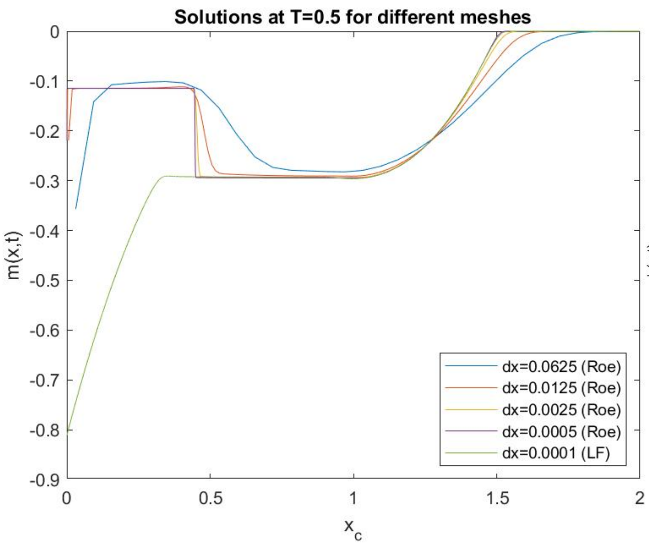

Figure 3.6: Numerical solutions of problem 1.4.c $m(x, t)$ for different mesh sizes and compared to the reference solution obtained using Lax-Friedrichs

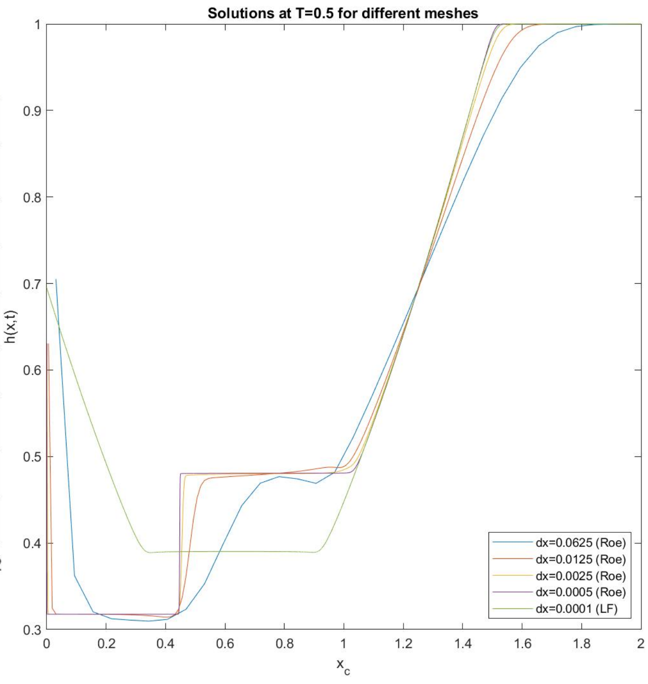

Figure 3.7: Numerical solutions of problem 1.4.c $h(x, t)$ for different mesh sizes and compared to the reference solution obtained using Lax-Friedrichs

A Matlab Codes

In this chapter, the Matlab programs for both the Lax-Friedrichs and the Roe methods are given.

Listing A.1: Lax-Friedrichs

```
%% Information
% Project 1: Lorenz Veithen and Mathieu Grondin
% Course: Numerical Methods for Consevation Laws
clc
clear ;
close all;
%% Problem Setup
Q1 = false; Q2a = false; Q2b = true; Q4 = false; decomposition=false;
r = 1;
inter = 0.001; counter = 0;
dxlist = [inter*r^4]; %inter*r^3 inter*r^2 inter*r];

for dx=dxlist
    x = 0:dx:2; xc = 0.5*dx:dx:2-0.5*dx; n = length(xc);
    CFL = 0.5;T = 2;
    uSource = 0.25; g = 1;

    % Initial Value Problem and Source Term
    if Q1; h0 =@(y)  1 + 0.5 * sin(pi*y); m0 =@(y) uSource*h0(y); end
    if Q2a; h0 =@(y) 1 - 0.1 * sin(pi*y); m0 =@(y) 0*(y>-1); end
    if Q2b; h0 =@(y) 1 - 0.2 * sin(2*pi*y); m0 =@(y) 0.5*(y>-1); end
    if Q4; h0 =@(y) 1*(y>-1); m0 =@(y) -1.5 .*(y<1) + 0 .*(y>1); end

    % Cell averages of initial value
    Discr_h0= zeros(1, n); Discr_m0 = zeros(1,n);
    for i=1:length(x)-1
        Discr_h0(1,i) = integral(h0, x(i), x(i+1))/dx;
        Discr_m0(1,i) = integral(m0, x(i), x(i+1))/dx;
        %disp(Discr_m0)
    end
    if Q4;Discr_h0(1,n) = Discr_h0(1,n-1); Discr_m0(1,n) = Discr_m0(1,n
       -1);
    else; Discr_h0(1,n) = Discr_h0(1,1); Discr_m0(1,n) = Discr_m0(1,1);
       end
    q0 = [Discr_h0; Discr_m0]; q_old = q0;

    % Time March
    t=0;
    while t < T
        k = CFL* dx/(max(abs(q_old(2,:)./q_old(1,:))+sqrt(g*q_old(1,:))))
            ;

```

```matlab
    if Q1; Sc_t = Sc(x, uSource, t, dx,g); else; Sc_t = zeros(2, n); end

    q_new = zeros(2,n); % For update
    for i=1:length(xc)

        % Necessary points and BC
        if (i==1)
            if Q4; q_il=q_old(:,1); %open BC
                else; q_il= q_old(:,n); %periodic BC
            end

        else ; q_il = q_old(:, i-1);
        end
        if (i==length(xc))
            if Q4; q_ir=q_old(:, n); %open BC
                else; q_ir=q_old(:,1);%periodic BC
            end

            else; q_ir = q_old(:,i+1);
        end
        q_i = q_old(:,i); % Current Position
        if decomposition

            % Matrix and diagonalisation
            u = q_i(2,1)/q_i(1,1); A = [0 1; -(u^2 - g*q_i(1,1)) 2*u];
            [vec, val] = eig(A); lambVec = [val(1,1); val(2,2)];
            S = vec; D = val; % S matrix , D lambda matrix

            % Decompose system and March with LF
            Vj = S\q_i; Vjl = S\q_il; Vjr = S\q_ir;
            Sc_t_ext = S\[Sc_t(1,i); Sc_t(2,i)];
            F_LF =@(u, v, lamb) 0.5*(lamb.*u + lamb.*v -(dx/k) *(v-u));
            V_new = Vj - (k/dx)*(F_LF(Vj, Vjr, lambVec) - ...
                F_LF(Vjl, Vj, lambVec)) + k.*Sc_t_ext;

            % Convert back to q coordinate
            q_new(:,i) = S*V_new;
        else
            f =@(q) [q(2,1); (q(2,1)^2 /q(1,1))+0.5*g*q(1,1)^2];
            F_LF =@(u, v) 0.5*(f(u) + f(v) -(dx/k) *(v-u));
            q_new(:,i) = q_i - (k/dx)*(F_LF(q_i, q_ir) ...
                - F_LF(q_il, q_i)) + k.*Sc_t(:,i);
        end
    end
    q_old = q_new;
    %plot(xc, q_old); pause(0.001)
    if counter > 100
        disp(t); counter = 0;
    end
    t = t + k; counter = counter + 1;
    end
    %figure(1); plot(xc, q_old(1,:)); hold on; figure(2);
    % plot(xc, q_old(2,:)); hold on
     plot(xc, q_old)
end



function res = Sc(xc, u, t, dx, g)
    n=length(xc); Sc1 = zeros(n, 1); Sc2 = Sc1;
    for i=1:n-1
        Sc1(i,1) = ((u-1)/(2*dx))*(sin(pi*(xc(i+1)-t)) -sin(pi*(xc(i)-t))
            );
        Sc2(i,1) = (1/dx) *(0.5*(g + (u-1)*u)*(sin(pi*(xc(i+1)-t)) ...
            - sin(pi*(xc(i)-t))) - 0.0625*g*(cos(2*pi*(xc(i+1)-t)) - ...
            cos(2*pi*(xc(i)-t))));
    end
    Sc1(n,1) = Sc1(1,1); Sc2(n,1) = Sc2(1,1);
    Source =[Sc1' ; Sc2']; res = Source;
end
```

Listing A.2: Roe Solver

```
%% Information - Roe Solver
% Project 1: Lorenz Veithen and Mathieu Grondin
% Course: Numerical Methods for Consevation Laws
clc
close all;
%% Problem Setup
r = 5;
inter = 0.0001;
dxlist = [inter*r^4 inter*r^3 inter*r^2 inter*r^1];
Err1=zeros(2, length(dxlist)); counter = 1;
M = zeros(2, length(0.5*dxlist(1):dxlist(1):2-0.5*dxlist(1)));

for dx=dxlist
    x = 0:dx:2; xc = 0.5*dx:dx:2-0.5*dx; n = length(xc);
    T =0.5;% The time step will change in the iteration
    uSource = 0.25; % Considered constant in this problem
    g = 1; % Gravitational constant ??

    %% - Initial conditions - add for other parts of the problem later
    h01 = zeros(n,1);h02a = zeros(n,1);h02b = zeros(n,1);
    for i=1:n-1
        %h01(i,1) = 1 - (0.5/(pi*dx))*(cos(pi*x(i+1)) - cos(pi*x(i)));%Q1
        %h02a(i,1)=1+ (0.1/(pi*dx))*(cos(pi*x(i+1)) - cos(pi*x(i))); %Q2a
        %h02b(i,1)=1+ (0.2/(2*pi*dx))*(cos(2*pi*x(i+1)) - cos(2*pi*x(i)))
            ;
    end
    %h01(n,1) = h01(1,1); h01 = h01'; m01 = uSource .* h01;
    %h02a(n,1) = h02a(1,1); h02a = h02a'; m02a = 0.* h02a;
    %h02b(n,1) = h02b(1,1); h02b = h02b'; m02b = 0.5*ones(1,n);
    h04=ones(1,n); m04=-1.5*(xc<1) +0*(xc>1);
    % - Source term cell averaged. Works
    Sc0 = Sc(x, uSource, 1, dx,g);

    %% Time loop
    t = 0;
    %h_old = h01; m_old = m01;
    %h_old = h02a; m_old = m02a;
    %h_old = h02b; m_old = m02b;
    h_old=h04; m_old = m04;
    q_old = [h_old; m_old];
    while t < T
        %Sc_t = Sc(x, uSource, t, dx,g);
        Sc_t = zeros(2, n); % Fpr exs 2
        q_new = zeros(2, n);
        %q_old_ext=[q_old(:,n) q_old q_old(:,1)]; %Periodic bc
        q_old_ext=[q_old(:,1) q_old  q_old(:,n)]; %Open bc
        qL_lamb = q_old_ext(:, 1:length(xc));
        qR_lamb = q_old_ext(:, 3:length(xc)+2);
        lambA = eigvalA(qL_lamb,qR_lamb,g);
        k =0.5*dx/lambA;
        for i=1:length(xc)
            qL = q_old_ext(:, i);
            qM = q_old_ext(:, i+1); qR = q_old_ext(:, i+2);
            A = matRoe(qL, qR, g);
            [vec, val] = eig(A);
            absval = abs(val);
            q_new(:, i) = update(absval, qL, qM, qR, k, dx, Sc_t(:, i));
        end
        q_old = q_new;
        plot(xc, q_old);
        pause(0.01);
        t = t + k;
        disp(t)
    end

    if counter ==1
            M(1, :) = q_old(1,:);
            M(2, :) = q_old(2,:);
    end

   %%Error computation
    %h_fi=M(1,1+r^(counter-1)*(0:length(xc)-1));
    %m_fi=M(2,1+r^(counter-1)*(0:length(xc)-1));
    %Err1(1, counter) =max(abs(q_old(1,:)-h_fi));
    %Err1(2, counter) = max(abs(q_old(2,:)-m_fi));
    %Err1(1, counter) =max(abs(q_old(1,:)-exact(xc, t)));
    %Err1(2, counter) = max(abs(q_old(2,:)-uSource*exact(xc,t)));
    counter = counter + 1;
end

function res = Sc(xc, u, t, dx, g)
    n=length(xc);
    S01a = zeros(n, 1);
    S01b = S01a;
    for i=1:n-1
        S01a(i,1)=((u-1)/(2*dx))*(sin(pi*(xc(i+1)-t)) -sin(pi*(xc(i)-t)))
            ;
        S01b(i,1)=(1/dx)*(0.5*(g+(u-1)*u)*(sin(pi*(xc(i+1)-t))...
            -sin(pi*(xc(i)-t)))-0.0625*g*(cos(2*pi*(xc(i+1)-t))...
            -cos(2*pi*(xc(i)-t))));
    end
    S01a(n,1) = S01a(1,1); S01b(n,1) = S01b(1,1);

    Source =[S01a' ; S01b']; res = Source;
end

function res = matRoe(qL, qR, g)
    zL = qL(1,1).^-0.5 .* qL; zR = qR(1,1).^-0.5 .* qR;
    z_avg=(zL+zR)/2;
    z_squared_avg=(zL.^2+zR.^2)/2;
    res = [0 1 ; ...
       (z_squared_avg(1)*g-z_avg(2)^2*z_avg(1)^-2) 2*z_avg(2)*z_avg(1)
           ^-1];
end

function res = update(absRoemat, qL,qM, qR, k, dx, Sm)
    f =@(q) [q(2,1); (q(2,1)^2 /q(1,1))+0.5*1*q(1,1)^2];
    F =@(QL, QR) (1/2) *(f(QR)+f(QL)) - (1/2) * absRoemat*(QR-QL);
    qM = qM - (k/dx) * (F(qM, qR) - F(qL, qM)) + k * Sm;
    res = qM;
end

function res= eigvalA(qL,qR,g)
    zL = qL(1,:).^-0.5 .* qL; zR = qR(1,:).^-0.5 .* qR;
    z_avg=(zL+zR)/2;
    z_squared_avg=(zL.^2+zR.^2)/2;
    a = (z_squared_avg(1,:).* g - z_avg(2,:).^2 .* z_avg(1,:).^-2);
    b =  2*z_avg(2,:).*z_avg(1,:).^-1;
    V = [(b + (b.^2 +4 * a).^0.5).*0.5 (b - (b.^2 +4 * a).^0.5).*0.5];
    res=max(V);
end
function res = exact(x,t)
    h0e =@(x) 1 + 0.5 * sin(pi*x);
    res = h0e(x-t);
end
```