SF2521: Homework Assignment 2

Thomas Garrett 920527-6133

Weicker David 940114-T115

7th March 2016

1 Stability of Numerical Schemes

We have the general scheme

$$\begin{aligned} U^{n+1} &= Q(t_n)U^n + \Delta t F^n \\ U^0 &= g \end{aligned}$$

where $U^n \in \mathbb{R}^d$.

1.1 Duhamel's Principle

We are given the following discrete Duhamel's Principle:

$$U^n = S_{\Delta t}(t_n, 0)g + \Delta t \sum_{\nu=0}^{n-1} S_{\Delta t}(t_n, t_{\nu+1})F^\nu, \tag{1}$$

where $t_n = n\Delta t$, and

$$\begin{aligned} S_{\Delta t}(t, t) &= I, \quad t \in \mathbb{R} \\ S_{\Delta t}(t_{n+1,t_\mu}) &= Q(t_n)S_{\Delta t}(t_n, t_\mu). \end{aligned}$$

We begin by showing that (1) holds by induction.

Base Case: $n = 0$

$$\begin{aligned} U^0 &= S_{\Delta t}(0, 0)g + \Delta t \sum_{\nu=0}^{-1} S_{\Delta t}(0, t_{\nu+1})F^\nu \\ &= S_{\Delta t}(0, 0)g + \Delta t 0 \\ &= g \end{aligned}$$

This is indeed true by the scheme given.

Now we assume that the discrete Duhamel's Principle fits the general scheme at step n, and we want to show that this implies that it fits for step $n + 1$.

$$\begin{aligned} U^{n+1} &= Q(t_n)U_n + F^n \\ &= Q(t_n)(S_{\Delta t}(t_n, 0)g + \Delta t \sum_{\nu=0}^{n-1} S_{\Delta t}(t_n, t_{\nu+1})F^\nu) + F^n \quad \text{by induction assumption} \\ &= Q(t_n)S_{\Delta t}(t_n, 0)g + \Delta t Q(t_n) \sum_{\nu=0}^{n-1} S_{\Delta t}(t_n, t_{\nu+1})F^\nu + S_{\Delta t}(t_{n+1}, t_{n+1})F^n \\ &= S_{\Delta t}(t_{n+1}, 0)g + \Delta t \sum_{\nu=0}^{n} S_{\Delta t}(t_{n+1}, t_{\nu+1})F^\nu \end{aligned}$$

So we have proven that Duhamel's principle holds for the given numerical scheme.

Duhamel's principle can give us an extra understanding on how an inhomogeneous term affects the solution. If $F^\nu = 0$ then the solution is given by :

$$U^n = S_{\Delta t}(t_n, 0)g$$

So the initial conditions g is affected by S and carried from 0 to t_n. If now, $F^\nu \neq 0$, we can see by Duhamel's principle that we can look at the inhomogeneous term F^ν as an initial condition at time t_ν. This initial condition is affected by the same S but now carried from t_ν to t_n. The solution is then given by the sum of all those "initial conditions" carried to t_n.

1.2 Bound in the Δt-norm

We now wish to show that

$$||S_{\Delta t}(t_{\nu+1}, t_\nu)||_{\Delta t} \leq Ke^{a\Delta t} \implies ||U^n||_{\Delta t} \leq K(e^{at_n}||g||_{\Delta t} + \int_0^{t_n} e^{a(t_n-s)}ds \max_{0\leq\nu\leq n-1} ||F^\nu||_{\Delta t}$$

Taking $||\cdot||_{\Delta t}$ of both sides of (1), then by Cauchy-Schwartz inequality, we have

$$\begin{aligned}
||U^n||_{\Delta t} &= ||S_{\Delta t}(t_n, 0)g + \Delta t \sum_{\nu=0}^{n-1} S_{\Delta t}(t_n, t_{\nu+1})F^\nu||_{\Delta t} \\
&\leq ||S_{\Delta t}(t_n, 0)||_{\Delta t}||g||_{\Delta t} + \Delta t \sum_{\nu=0}^{n-1} ||S_{\Delta t}(t_n, t_{\nu+1})||_{\Delta t}||F^\nu||_{\Delta t} \\
&\leq Ke^{at_n}||g||_{\Delta t} + \Delta t \sum_{\nu=0}^{n-1} ||S_{\Delta t}(t_n, t_{\nu+1})||_{\Delta t}||F^\nu||_{\Delta t} \\
&\leq Ke^{at_n}||g||_{\Delta t} + \Delta t \sum_{\nu=0}^{n-1} Ke^{\alpha(t_n-t_{\nu+1})}||F^\nu||_{\Delta t}
\end{aligned}$$

We can assume that α is positive because otherwise increasing Δt will decrease the bound.

We notice that $\Delta t \sum_{\nu=0}^{n-1} Ke^{a(t_n-t_{\nu+1})}$ is a right Remman sum of a strictly decreasing function, thus

$$\begin{aligned}
&\leq Ke^{at_n}||g||_{\Delta t} + K\int_0^{t_n} e^{\alpha(t_n-s)}ds||F^\nu||_{\Delta t} \\
&\leq K(e^{at_n}||g||_{\Delta t} + \int_0^{t_n} e^{\alpha(t_n-s)}ds \max_{0\leq\nu\leq n-1} ||F^\nu||_{\Delta t})
\end{aligned}$$

1.3 α Value

If $a = \Delta t^{-1/2}$, then we would have $||S_{\Delta t}(t_{\nu+1}, t_\nu)||_{\Delta t} \leq Ke^{\sqrt{\Delta t}}$. Plugging this into our second inequality, we obtain,

$$||U^n||_{\Delta t} \leq K(e^{n\sqrt{t_n}}||g||_{\Delta t} + \int_0^{t_n} e^{n\sqrt{t_n-s}}ds \max_{0\leq\nu\leq n-1} ||F^\nu||_{\Delta t})$$

That means that for a given time t_n, if we increase the number of time steps n, the bound increases exponentially. Since it is always possible to increase n, that means that we have no bound at all. So we can no longer conclude that we have stability.

2 Shallow water model

In this section, we are going to solve numerically the time-dependent shallow water model in one spatial dimension. At first, we have solid walls at the boundary and thus the waves should be reflected. We will also linearize this non linear model and compare the solution of the linear problem with that of the non linear one. Finally, we will impose non-reflecting boundary conditions.

The shallow water model is given by :

$$\begin{aligned} h_t + (hv)_x &= 0 \\ (hv)_t + (hv^2 + \frac{1}{2}gh^2)_x &= 0 \\ \text{on } (x,t) &\in [0,L] \times [0,\infty) \end{aligned}$$

The initial condition is given by :

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

Regarding the boundary condition, we imposed solid walls on $x = 0$ and $x = L$. We will see in the next section how that is implemented in practice.

2.1 Numerical solution

To solve this problem numerically, we implemented a finite volume scheme using the Lax-Friedrichs method. The matlab code is available at the end of the report.

It is a regular finite volume scheme. However, we needed to impose the solid walls boundary condition. This is done, as explained in Leveque chapter 7, by adding two ghosts cells (Q_0 and Q_{N+1}) containing each h and hv and saying :

$$\begin{aligned} h_0 &= h_1 \\ h_N &= h_{N+1} \\ v_0 = -v_1 &\implies hv_0 = -hv_1 \\ v_N = -v_{N+1} &\implies hv_N = -hv_{N+1} \end{aligned}$$

This will impose an axial symmetry with respect to the boundary and impose the solid walls condition.

Figure 1 shows the results for four seconds and $\frac{\Delta t}{\Delta x} = 0.3$. We can see the initial condition as well as the reflection of the waves of the boundaries. We can also see that when the two waves meets around $t = 3$, they loose height. However, there is no numerical dissipation since the total height is conserved (this has been checked!).

Figure 2 shows the same results but in a 2D plot. We can see the waves propagate and collide. We can also note that the side of the wave that is "in front", meaning on the side where the wave goes, is steeper than the other. This edge becomes also even steeper after the collision. We can recognize here a characteristic of non linear hyperbolic PDE's, where discontinuities can appear.

We will now run the program with a larger ϵ. Previously the value was 0.1, we will now try with $0.4, 0.8, 1.2$. We can first note that we have to decrease the ratio $\frac{\Delta t}{\Delta x}$ if we increase ϵ. This is because increasing ϵ also increases the derivative of the height and thus the flux. Thus, to keep a stable numerical scheme, we have to decrease α.

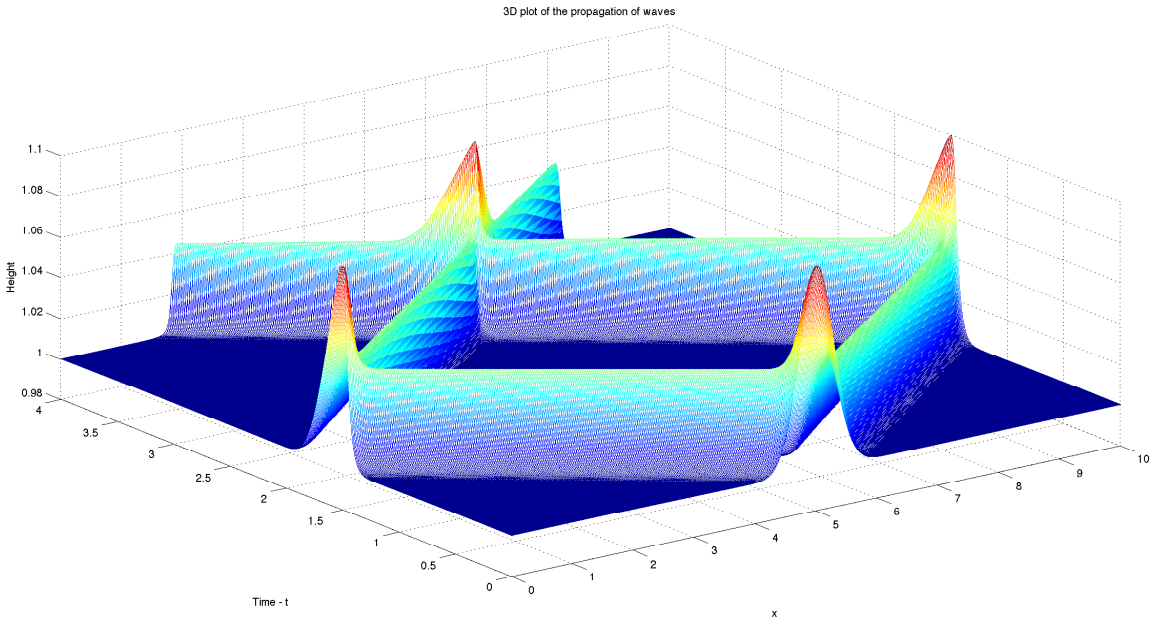

Figure 1: Results for the non linear system of PDE

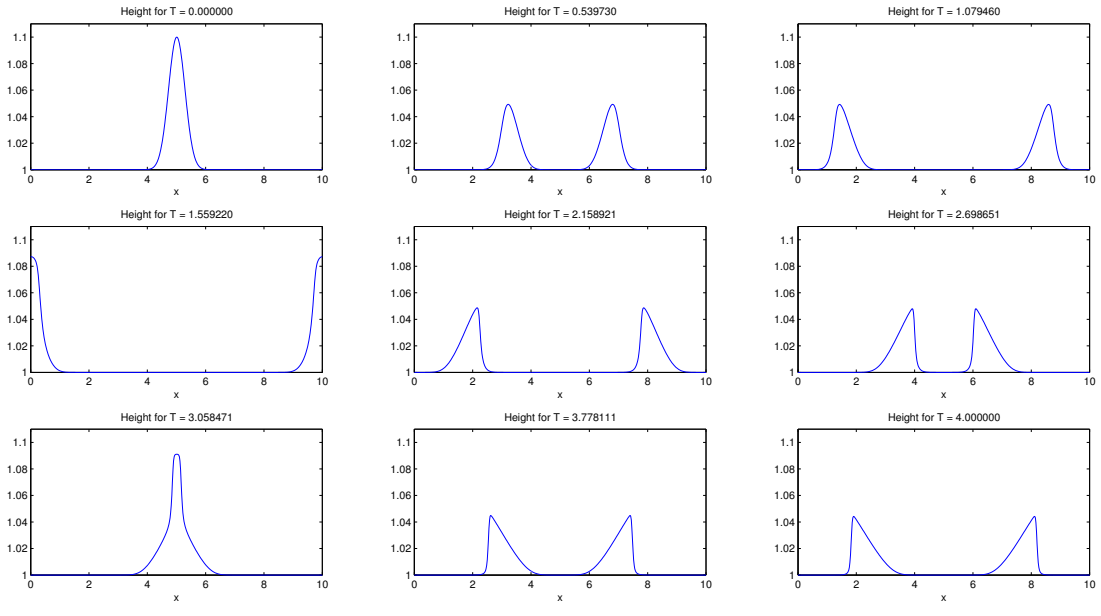

Figure 2: Results for the non linear system of PDE

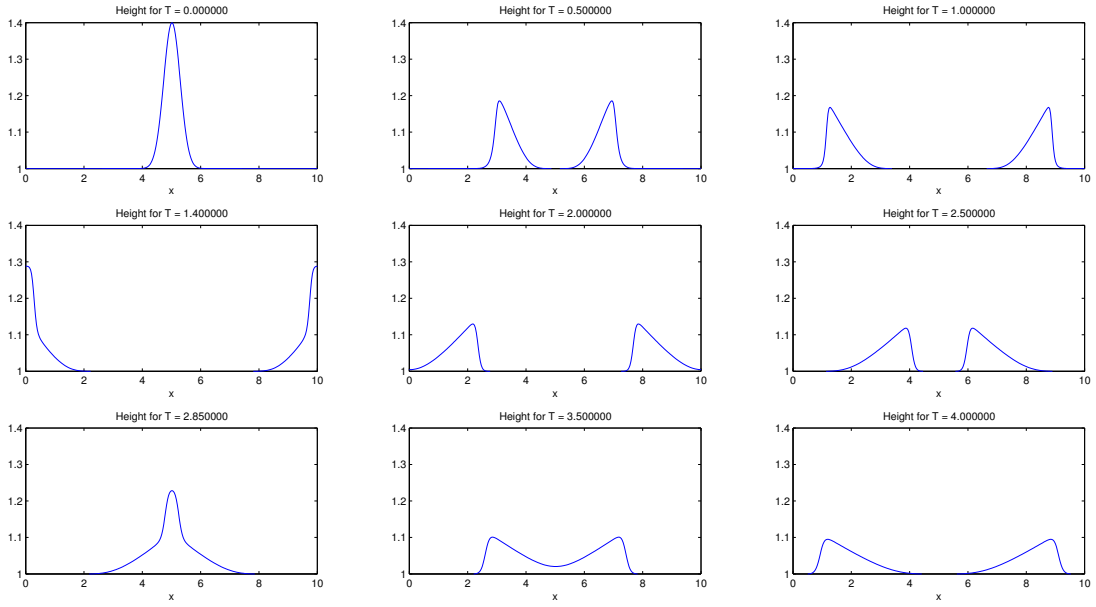

Figure 3: $\epsilon = 0.4$ and $\frac{\Delta t}{\Delta x} = 0.25$

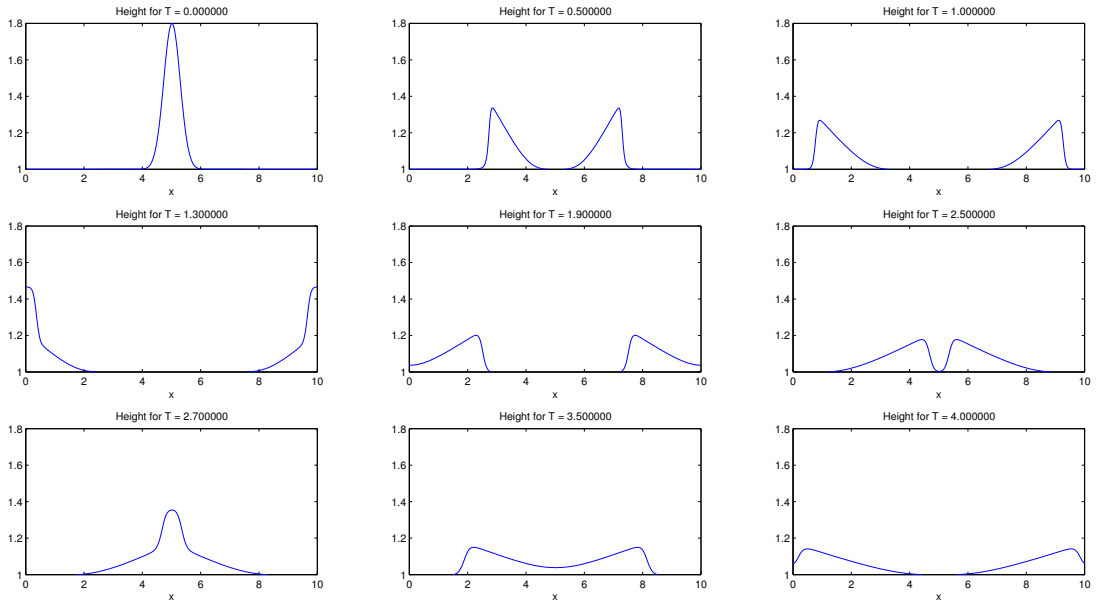

Figure 4: $\epsilon = 0.8$ and $\frac{\Delta t}{\Delta x} = 0.21$

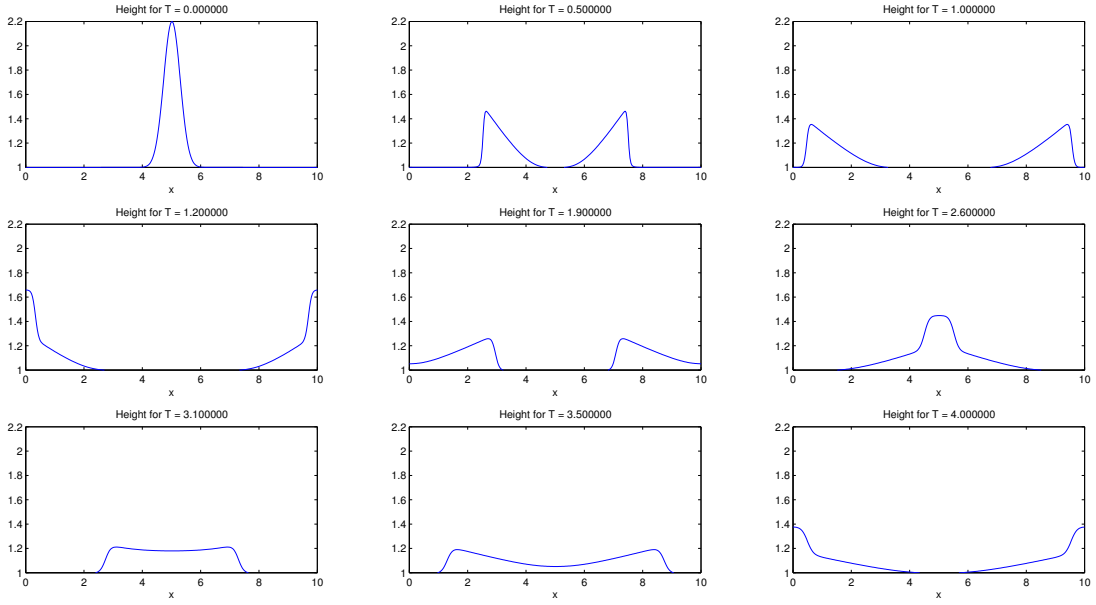

Figure 5: $\epsilon = 1.2$ and $\frac{\Delta t}{\Delta x} = 0.19$

Figures 3, 4, 5 shows the results for $\epsilon = 0.4, 0.8, 1.2$ respectively.

Let us look at the differences between the solutions. There is the amplitude obviously. Because the initial condition is "higher" with a larger ϵ, then so is the height of the propagated wave. We can note, however, that the larger the ϵ the quicker the wave looses its amplitude. This is intuitive. Indeed, a high thin wave will want to be flatter.

We can also say that the wave speed increases with ϵ. The time needed to reach the boundary is around 1.6 for $\epsilon = 0.1$ but it is 1.3 for $\epsilon = 0.8$ for example. And for the last one, at final time, the waves are already on the other boundary! This is also intuitive that higher waves are quicker.

The collisions look also to affect more the wave. The higher the ϵ, the flatter after each collision the waves look.

Finally, we are going to change the factor α in the numerical scheme. Figure 6 shows results after half a second when changing the parameter C_0. We kept the ration $\frac{\Delta t}{\Delta x}$ at 0.3. We can comment that increasing α seems to decrease the amplitude of the wave while making it wider. The mean speed however is not changed.

2.2 Linearization

The shallow water equation can be written in quasilinear form as

$$u_t + f'(u)u_x = 0$$

where $u = (h, hv)^T$,

$$f'(u) = \begin{pmatrix} 0 & 1 \\ -(\frac{u_2}{u_1})^2 + gu_1 & 2(\frac{u_2}{u_1}) \end{pmatrix}$$

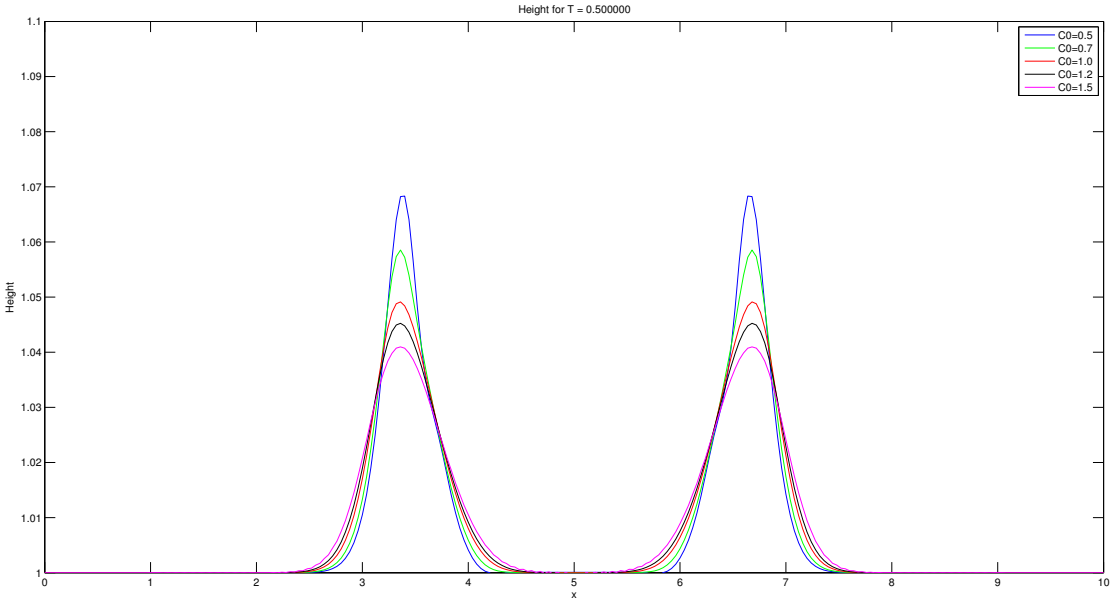

Figure 6: Results when changing α

and

$$q(x,0) = \begin{pmatrix} H + \epsilon e^{-(x-L/2)^2/w^2} \\ 0 \end{pmatrix}$$

Now, to make this system linear we must simply pick a constant state (h_0, v_0) which is consistent with the boundary and initial conditions. We choose (h_0, v_0) at $x = 0$ and $t = 0$. We can now compute (h_0, v_0) using the initial condition. We get $(h_0, v_0) \approx (1, 0)$. Thus we have

$$f'(u) = \begin{pmatrix} 0 & 1 \\ g & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 9.61 & 0 \end{pmatrix}$$

We can see easily $f'(u)$ has eigenvalues $\lambda_{1,2} = \pm\sqrt{9.61} = \pm 3.1$ which are real. It also has a full set of eigenvectors i.e. $(1, 3.1)^T$ and $(-1, 3.1)$. This together confirm that the linear problem is hyperbolic. We also know that the wave speeds are the eigenvalues, so we have wave speeds $\pm\sqrt{9.61}$

2.2.1 Analytical Solution

We have the PDE,

$$q_t + f'(u)q_x = 0$$

which we have shown can be written as

$$\begin{aligned} & q_t + VDV^{-1}q_x = 0 \\ \implies & V^{-1}q_t + DV^{-1}q_x = 0 \end{aligned}$$

where

$$V = \begin{pmatrix} 1 & -1 \\ 3.1 & 3.1 \end{pmatrix}, \quad D = \begin{pmatrix} 3.1 & 0 \\ 0 & -3.1 \end{pmatrix}, \quad \text{and } V^{-1} = \begin{pmatrix} 0.5 & 0.1613 \\ -0.5 & 0.1613 \end{pmatrix}$$

and initial condition

$$q(x,0) = \begin{pmatrix} H + \epsilon e^{-(x-L/2)^2/w^2} \\ 0 \end{pmatrix}, \qquad 0 \le x \le L$$

Now, defining a new variable $r = V^{-1}q$ we have the decoupled system of equations

$$r_t + Dr_x = 0$$

and

$$r(x,0) = V^{-1}q(x,0) = \frac{1}{2}\begin{pmatrix} H + \epsilon e^{-(x-L/2)^2/w^2} \\ -H - \epsilon e^{-(x-L/2)^2/w^2} \end{pmatrix}$$

From Lavengen, we know the solutions of this system are

$$r_1(x,t) = r_1(x + \lambda_1 t, 0) = \frac{1}{2}(H + \epsilon e^{-(x+3.1t-L/2)^2/w^2})$$
$$\text{and} \quad r_2(x,t) = r_2(x + \lambda_2 t, 0) = \frac{1}{2}(-H - \epsilon e^{-(x-3.1t-L/2)^2/w^2})$$

Finally switching back to q, we get

$$\begin{aligned} q(x,t) = Vr(x,t) &= \frac{1}{2}\begin{pmatrix} H + \epsilon e^{-(x+3.1t-L/2)^2/w^2} + H + \epsilon e^{-(x-3.1t-L/2)^2/w^2} \\ 3.1(H + \epsilon e^{-(x+3.1t-L/2)^2/w^2} - H - \epsilon e^{-(x-3.1t-L/2)^2/w^2}) \end{pmatrix} \\ &= \frac{1}{2}\begin{pmatrix} 2H + \epsilon e^{-(x+3.1t-L/2)^2/w^2} + \epsilon e^{-(x-3.1t-L/2)^2/w^2} \\ 3.1(\epsilon e^{-(x+3.1t-L/2)^2/w^2} - \epsilon e^{-(x-3.1t-L/2)^2/w^2}) \end{pmatrix} \end{aligned}$$

Now, simply plugging in $t = 1$ we get

$$q(x,1) = \frac{1}{2}\begin{pmatrix} 2H + \epsilon e^{-(x+3.1-L/2)^2/w^2} + \epsilon e^{-(x-3.1-L/2)^2/w^2} \\ 3.1(\epsilon e^{-(x+3.1-L/2)^2/w^2} - \epsilon e^{-(x-3.1-L/2)^2/w^2}) \end{pmatrix}$$

which is plotted in 7. We note that this solution does take into account any boundary conditions, and that the waves will not be reflected. However the waves have not hit the walls until after $T = 1$ so our analytical solution still holds at $T = 1$.

2.2.2 Numerical Solution of the Linear Problem

To compare the results from the non-linear and linear problems, we compare the numerical solutions of the problems using the same method and the proper boundary conditions. The code for the numerical solution of the linear problem is almost exactly the same as for the linear problem, except the flux f. For the linear problem we compute the flux function from the information above.

$$f'(u) = \begin{pmatrix} 0 & 1 \\ g & 0 \end{pmatrix} \rightarrow f(u) = \begin{pmatrix} u_2 \\ gu_1 \end{pmatrix}$$

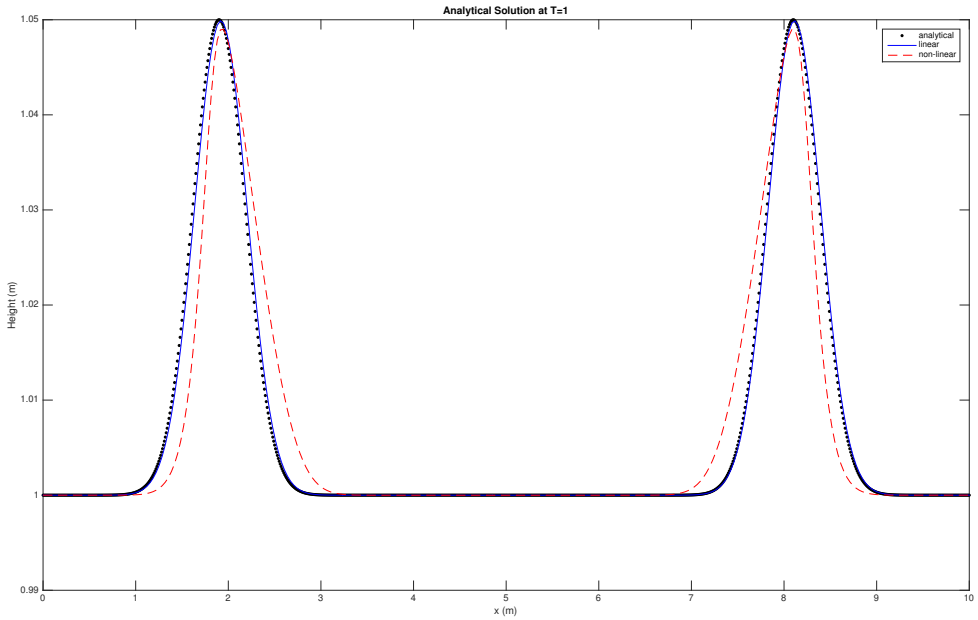

Figure 7: Analytical Solution of the Linear Problem

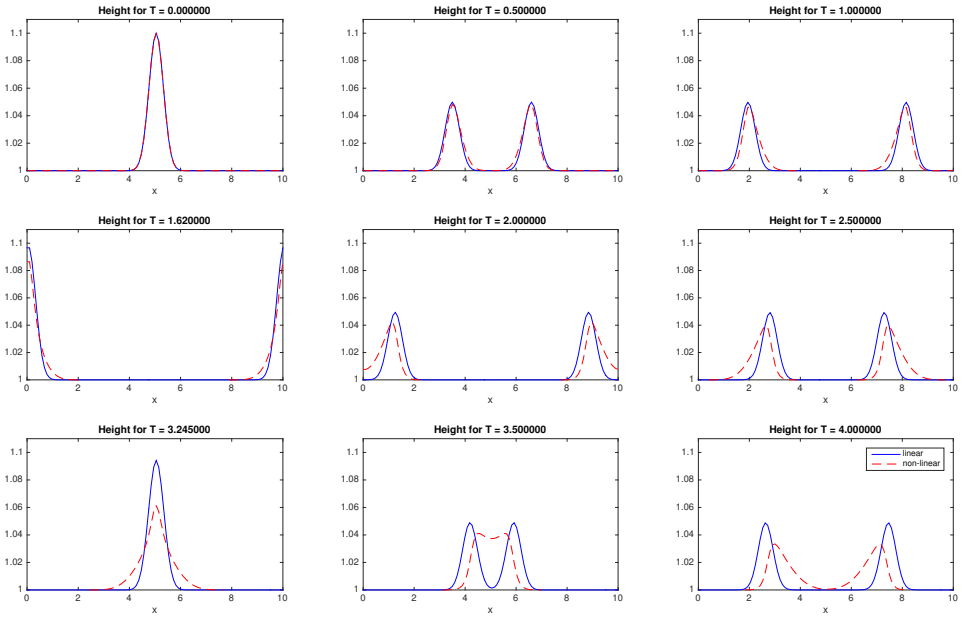

Figure 8: Comparison of Linear and Non-linear Problems

We plot the linear and non-linear numerical solutions in 8 so they can be compared on the same plots. The first thing we can see is that the waves from the linear solution are totally symmetrical. This is in contrast to the non-linear problem, in which the waves are already noticeably unsymmetrical at $T = 1$. The second difference that we notice is that the waves in the linear case retain their shape (same width and height) before and after collisions. In the non-linear case, this is not true. One can see that after a collision, the waves in the non-linear solution become shorter and wider. We can also see that the linear waves move quicker than the non-linear waves.

2.3 Non-reflecting boundary conditions

Finally, we are going to apply non reflecting conditions at the boundaries. As explained in Leveque chapter 7, we extrapolate the variables at the ghost cells by saying :

$$\begin{aligned} h_0 &= h_1 \\ h_N &= h_{N+1} \\ v_0 = v_1 &\implies hv_0 = hv_1 \\ v_N = v_{N+1} &\implies hv_N = hv_{N+1} \end{aligned}$$

Intuitively, the flow is going to "go out" of the domain with such boundary conditions.

The code is available at the end of the report. The structure does not change much, the only difference is the computations of the values of the ghost cells.

Figure 9 shows the results when applying the extrapolation for the ghost cells. We can see that at first nothing changes from the original problem. This is coherent since we only changed the boundary conditions . But the waves go out of the domain when reaching the boundaries. This is exactly what we wanted ! There is no reflection noticeable. To investigate a bit more, we can compute the integral of the height on the domain. At $t = 2.1$, this value is 10 with accuracy up to floating point errors so we can be sure that there is no reflection at all.

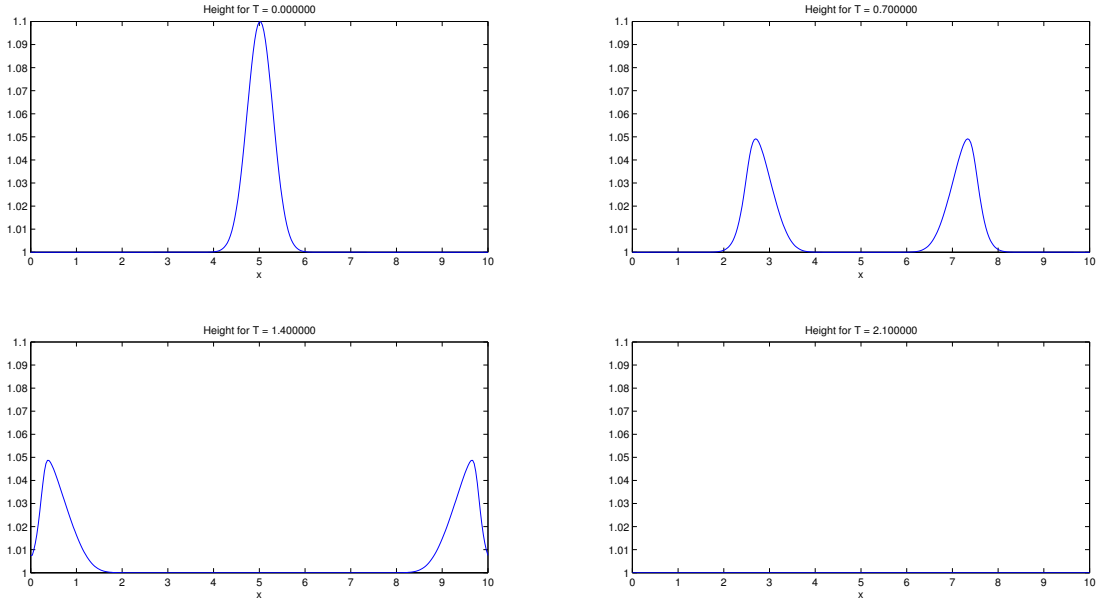

Figure 9: Results with non reflective boundary conditions

3 Matlab codes

```
function [Q,x,t,cons] = adv(xSteps,ratio,Tend,alpha,epsilon)
%Problem 2.1 Lax-Friedrich method for non linear problem

%given variables
L = 10;
H = 1;
g = 9.61;
w = 0.4;
%calculate number of steps
dx = L/xSteps;
dt = ratio*dx;
tSteps = round(Tend/dt);
%flux function
f = @(u) [ u(2) , u(2)^2./u(1) + 0.5*g*u(1).^2];
%Lax-Friedrich flux function
FLxF = @(u2,u1) (0.5*(f(u2)+f(u1) - alpha*dx/dt*(u2-u1)));

%Initial Conditions
Q = zeros(xSteps+2, 2*(tSteps+1));
Q(2:(end-1),1) = H+epsilon*exp(-((dx/2:dx:(L-dx/2)) - L/2).^2/w^2);

F = zeros(xSteps+1,2);
for i = 1:tSteps+1
    % Ghost point values
    Q(1,2*i-1) = Q(2,2*i-1);
    Q(1,2*i) = -Q(2,2*i);
    Q(end,2*i-1) = Q(end-1,2*i-1);
    Q(end,2*i) = -Q(end-1,2*i);

    for j = 1:xSteps+1
        F(j,:) = FLxF( Q(j+1,(2*i-1):(2*i)), Q(j,(2*i-1):(2*i)) );
```

```
    end
    for j = 2:xSteps+1
        Q(j,2*i+1:2*i+2) = Q(j,2*i-1:2*i) - dt/dx * (F(j,:)-F(j-1,:));
    end
end
x = linspace(0,L,xSteps+1);
t = linspace(0,Tend,tSteps+1);
Q = Q(1:end-1,2*(1:tSteps+1)-1);
cons = sum(Q(:,:))/(xSteps+1);
end
```

```
function [Q,x,t,cons] = advLinear(xSteps,ratio,Tend)
%Problem 2.1 Lax-Friedrich method for linear problem

%given variables
L = 10;
H = 1;
g = 9.61;
w = 0.4;
epsilon = 0.1;
%calculate number of steps
dx = L/xSteps;
dt = ratio*dx;
tSteps = round(Tend/dt);
%flux function
f = @(u) [u(2) , g*u(1)];
%Lax-Friedrich flux function
FLxF = @(u2,u1) (0.5*(f(u2)+f(u1) - dx/dt*(u2-u1)));

%Initial Conditions
Q = zeros(xSteps+2, 2*(tSteps+1));
Q(2:(end-1),1) = H+epsilon*exp(-((dx/2:dx:(L-dx/2)) - L/2).^2/w^2);

F = zeros(xSteps+1,2);
for i = 1:tSteps+1
    % Ghost point values
    Q(1,2*i-1) = Q(2,2*i-1);
    Q(1,2*i) = -Q(2,2*i);
    Q(end,2*i-1) = Q(end-1,2*i-1);
    Q(end,2*i) = -Q(end-1,2*i);

    for j = 1:xSteps+1
        F(j,:) = FLxF( Q(j+1,(2*i-1):(2*i)), Q(j,(2*i-1):(2*i)) );
    end
    for j = 2:xSteps+1
        Q(j,2*i+1:2*i+2) = Q(j,2*i-1:2*i) - dt/dx * (F(j,:)-F(j-1,:));
    end
end
%close all;
x = linspace(0,L,xSteps+1);
t = linspace(0,Tend,tSteps+1);
Q = Q(1:end-1,2*(1:tSteps+1)-1);
cons = sum(Q(:,:))/(xSteps+1);
%mesh(t,x,Q);%Q(:,2*(1:tSteps+1)-1))
xlabel('Time (s)')
ylabel('x (m)')
zlabel('Height (m)')
%mesh(0:dx:L, (0:dt:T)', Q(:,2*(1:tSteps+1)-1))
rotate3d on
%cons = sum(Q(2:xSteps+1,2*(1:tSteps+1)-1));
end
```

```
function [Q,x,t,cons] = noReflex(xSteps,ratio,Tend)
%Problem 2.1 Lax-Friedrich method with non reflexive conditions
close all;
%given variables
```

```
L = 10;
H = 1;
g = 9.61;
w = 0.4;
epsilon = 0.1;
%calculate number of steps
dx = L/xSteps;
dt = ratio*dx;
tSteps = round(Tend/dt);
%flux function
f = @(u) [ u(2) , u(2)^2./u(1) + 0.5*g*u(1).^2];
%Lax-Friedrich flux function
FLxF = @(u2,u1) (0.5*(f(u2)+f(u1) - dx/dt*(u2-u1)));

%Initial Conditions
Q = zeros(xSteps+2, 2*(tSteps+1));
Q(2:(end-1),1) = H+epsilon*exp(-((dx/2:dx:(L-dx/2)) - L/2).^2/w^2);

F = zeros(xSteps+1,2);
for i = 1:tSteps+1
    % Ghost point values
    Q(1,2*i-1:2*i) = Q(2,2*i-1:2*i);
    Q(end,2*i-1:2*i) = Q(end-1,2*i-1:2*i);

    for j = 1:xSteps+1
        F(j,:) = FLxF( Q(j+1,(2*i-1):(2*i)), Q(j,(2*i-1):(2*i)) );
    end
    for j = 2:xSteps+1
        Q(j,2*i+1:2*i+2) = Q(j,2*i-1:2*i) - dt/dx * (F(j,:)-F(j-1,:));
    end
end
x = linspace(0,L,xSteps+1);
t = linspace(0,Tend,tSteps+1);
Q = Q(1:end-1,2*(1:tSteps+1)-1);
cons = sum(Q(:,:))/(xSteps+1);
end
```