SF2520 - Lab 6

Goyens Florentin & Weicker David

January 2016

Part 1

For the model hyperbolic problem

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0 \tag{1}$$

with $a > 0$, $0 < x < 2$ and $t > 0$. We consider the initial condition $u(x, 0) = 0$ for $0 < x \leq 2$. Since $a > 0$ the solution will travel towards the right and we need a boundary condition on the left-hand side of the domain. Take the square signal

$$u(0,t) = \begin{cases} 1 & -(n+1)T/2 < t \leq -nT/2 \quad n = 0, 2, 4 \dots \\ -1 & -(n+1)T/2 < t \leq -nT/2 \quad n = 1, 3, 5 \dots \end{cases}$$

where T is the period time.

The point of interest is to compare the behaviour of three methods 1) upwind, 2) Lax-Friedrich and 3) Lax-Wendroff. Let's first define the following methods.

1) Upwind FTBS The upwind method uses forward time and backward space schemes. So we get the first order in space and time

$$u_{i,k+1} = (1 - \sigma)u_{i,k} + \sigma u_{i-1,k}.$$

2) Lax-Friedrich method is defined by

$$u_{i,k+1} = \frac{u_{i-1,k} + u_{i+1,k}}{2} - \frac{\sigma}{2}(u_{i+1,k} - u_{i-1,k}).$$

3) Lax-Wendroff method is defined by

$$u_{i,k+1} = u_{i,k} - \frac{\sigma}{2}(u_{i+1,k} - u_{i-1,k}) + \frac{\sigma^2}{2}(u_{i+1,k} - 2u_{i,k} + u_{i-1,k}).$$

For the last two schemes, the value $u_{end,k+1}$ is not well defined. We use linear extrapolation

$$u_{end,k+1} = 2u_{end-1,k} - u_{end-2,k}$$

.

We look at the results on the test problem for the values $\sigma = \{0.8, 1, 1.1\}$. It is clear on figure 1 that $\sigma = 1.1$ gives an unstable scheme for each method. And the "magic" $\sigma = 1$ transports the solution as it should. The interesting points to analyse are in the case $\sigma = 0.8$ where different behaviour occur. The upwind and Lax-Friedrich methods introduce a smoothing. This is an illustration of an artificial dissipation. Finally, the Lax-Wendroff generates oscillations, this is to be expected for a method that is second order in time and the errors come from the dispersion in the numerical method.

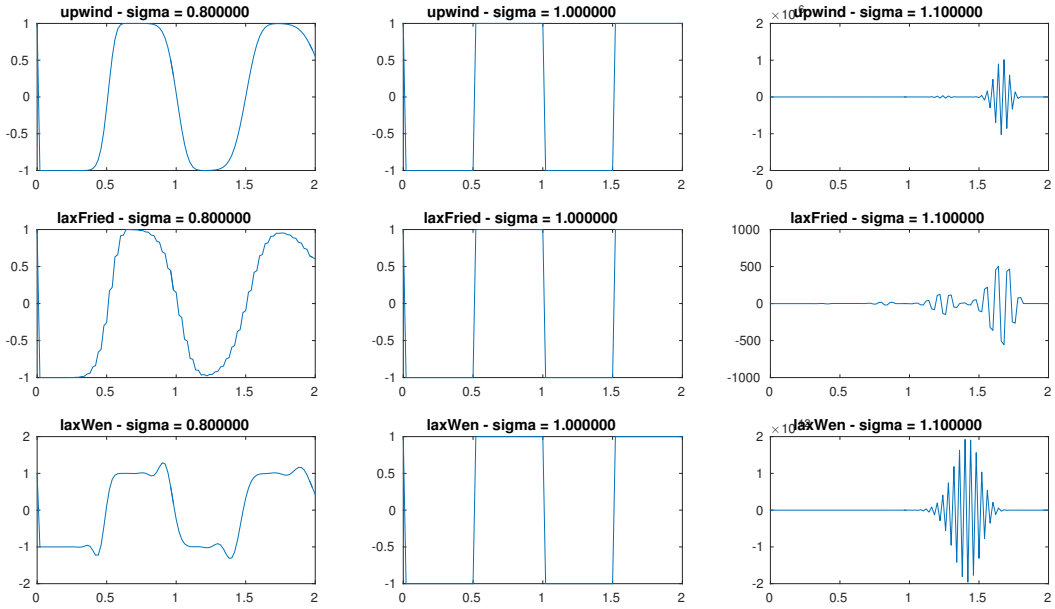

Figure 1: Results for the different methods

Part 2

We are now concerned with the example 8.1 and we will try different methods to solve it.

a) Upwind method

We are first going to use the upwind method to solve the problem. The first step is to discretize the equation. The PDE is :

$$\frac{\partial T}{\partial t} + v\frac{\partial T}{\partial x} + a(T - T_{cool}) = 0$$

Using forward time and backward space differences and denoting $T_{i,k} \approx T(ih_x, kh_t)$, we get :

$$\frac{T_{i,k+1} - T_{i,k}}{h_t} + v\frac{T_{i,k} - T_{i-1,k}}{h_x} + a(T_{i,k} - T_{cool}) = 0$$
$$T_{i,k+1} = (1 - \frac{vh_t}{hx} - ah_t)T_{i,k} + \frac{vh_t}{h_x}T_{i-1,k} + ah_tT_{cool}$$

Because we have an initial condition and a boundary condition for $x = 0$, we problem is well posed. The implementation is done in the code *temp.m*. Figure 2 presents the results for the implementation of the upwind method for the given problem.

The figure presents what is to be expected. At $x = 0$, we can see the given boundary condition. We also see that the temperature propagates along the x-axis but that there are some dissipation and that the mean temperature decreases as we go along the x-axis.

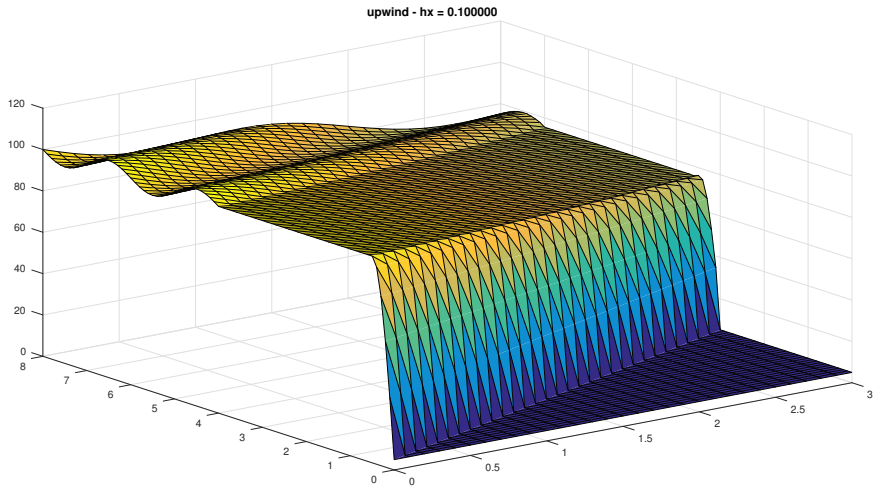

Figure 2: Solution with upwind method

b) Derivation of the Lax-Wendroff rule

In this part, we are going to try to compute a Lax-Wendroff rule for the problem. The trick is to use the PDE to express the time derivatives with spacial derivatives and then use Taylor expansion. So, in this instance, we have :

$$\frac{\partial T}{\partial t} = -v\frac{\partial T}{\partial x} - a(T - T_{cool})$$

So the second order derivative, we have :

$$\begin{aligned}
\frac{\partial^2 T}{\partial t^2} &= \frac{\partial}{\partial t}(-v\frac{\partial T}{\partial x} - a(T - T_{cool})) \\
&= -v\frac{\partial}{\partial x}(\frac{\partial T}{\partial t}) - a\frac{\partial T}{\partial t} \\
&= -v\frac{\partial}{\partial x}(-v\frac{\partial T}{\partial x} - a(T - T_{cool})) - a(-v\frac{\partial T}{\partial x} - a(T - T_{cool})) \\
&= v^2\frac{\partial^2 T}{\partial x^2} + 2av\frac{\partial T}{\partial x} + a^2(T - T_{cool})
\end{aligned}$$

Now we use Taylor expansion up until the second order :

$$\begin{aligned}
T_{i,k+1} &= T_{i,k} + h_t\frac{\partial T}{\partial t} + \frac{h_t^2}{2}\frac{\partial^2 T}{\partial t^2} \\
&= T_{i,k} + h_t(-v\frac{\partial T}{\partial x} - a(T - T_{cool})) + \frac{h_t^2}{2}(v^2\frac{\partial^2 T}{\partial x^2} + 2av\frac{\partial T}{\partial x} + a^2(T - T_{cool}))
\end{aligned}$$

Using central differences for spatial derivatives, we get :

$$T_{i,k+1} = c_1 T_{i-1,k} + c_2 T_{i,k} + c_3 T_{i+1,k} + c_4$$

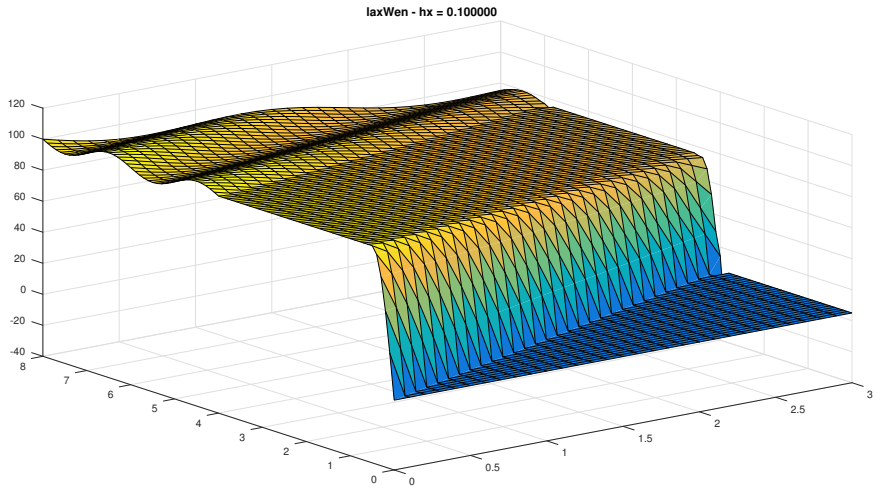

Figure 3: Solution with Lax-Wendroff method

Where the c_i are defined by :

$$c_1 = \frac{h_t v}{2h_x} + \frac{h_t^2 v^2}{2h_x^2} - \frac{avh_t^2}{2h_x}$$
$$c_2 = 1 - ah_t + \frac{a^2 h_t^2}{2} - \frac{v^2 h_t^2}{h_x^2}$$
$$c_3 = -\frac{vh_t}{2h_x} + \frac{v^2 h_t^2}{2h^2} + \frac{avh_t^2}{2h_x}$$
$$c_4 = (ah_t - \frac{a^2 h_t^2}{2})T_{cool}$$

c) Presentation of the results

In this final part, we are going to use the Lax-Wendroff formula derived in the previous part. We have first to notice that because we need both the point to the right and the point to the left, we will have to use the extrapolation formula already used in the first part for the right end point (that is for $x = L$). That is :

$$T_{end,k} = 2T_{end-1,k} - T_{end-2,k}$$

Using this and the boundary condition given, we can implement the problem. This is also done in *temp.m*. Figure 3 presents the results.

We can see that this plot is very similar to the one obtained with the upwind method. All the remarks done previously thus hold. The only notable difference is that we can see a peak due to the extrapolation formula on the far right end side (as shown on figure 4).

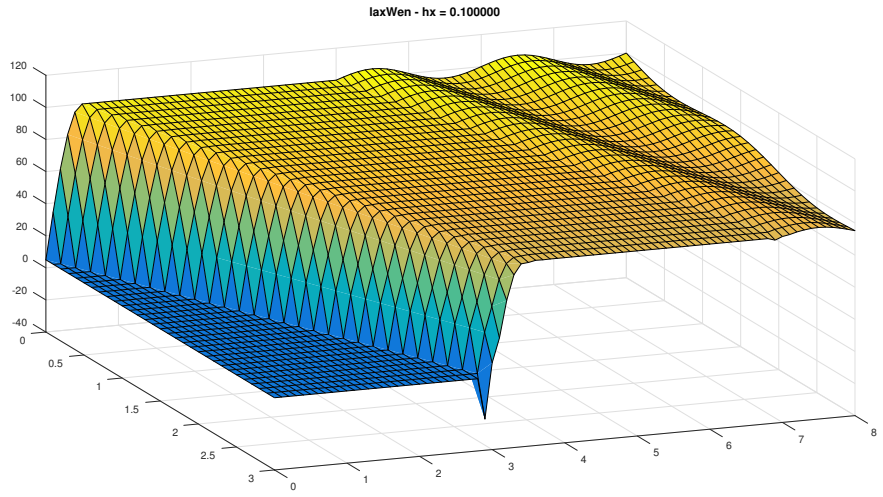

Figure 4: Oscillation due to extrapolation

Matlab codes

```
function [] = transport()
%TRANSPORT
close all;
clear all;

T = 1;
a = 1;
N = 100;

hx = 2/N;
x = linspace(0,2,N+1);
sigma = [0.8 1 1.1];
methods = {@upwind,@laxFried,@laxWen};
figure;
for i = 1:3
    meth = methods{i};
    for j = 1:3
        sig =sigma(j);
        ht = hx*sig/a;
        t = 0:ht:2*T;
        m = length(t);
        u = meth(t,sig,N,m);
        subplot(3,3,3*(i-1)+j);
        str = func2str(meth);
        plot(x,u(end,:));title(sprintf('%s - sigma = %f',str,sig));

    end
end
end

function u = upwind(t,sigma,N,m)
%Solve the prolem with upwind method
u = zeros(m,N+1);
u(1,1) = bcLeft(t(1));
for k = 1:m-1
    u(k+1,1) = bcLeft(t(k+1));
    u(k+1,2:end) = (1-sigma)*u(k,2:end) + sigma*u(k,1:end-1);
end
```

```
end

function u = laxFried(t,sigma,N,m)
%Solve the prolem with Lax-Friendrich
u = zeros(m,N+1);
u(1,1) = bcLeft(t(1));
for k = 1:m-1
    u(k+1,1) = bcLeft(t(k+1));
    u(k+1,2:end-1) = (u(k,3:end)+u(k,1:end-2))/2 - sigma*(u(k,3:end)-u(k,1:end-2))/2;
    u(k+1,end) = 2*u(k+1,end-1)-u(k+1,end-2);
end
end

function u = laxWen(t,sigma,N,m)
%Solve the prolem with Lax-Wendroff
u = zeros(m,N+1);
u(1,1) = bcLeft(t(1));
for k = 1:m-1
    u(k+1,1) = bcLeft(t(k+1));
    u(k+1,2:end-1) = u(k,2:end-1) - sigma*(u(k,3:end)-u(k,1:end-2))/2 + sigma^2*(u(k↩
        ,3:end)-2*u(k,2:end-1)+u(k,1:end-2))/2;
    u(k+1,end) = 2*u(k+1,end-1)-u(k+1,end-2);
end
end

function [uLeft] = bcLeft(t)
%Gives the left boundary condition
T = 1;
uLeft = (-1).^floor(2.*t./T);
end
```

```
function [T] = temp(hx,tend)
%TEMP
close all;

a = 0.1;
v = 1;
L = 3;
Tcool = 5;
Thot = 100;

x = 0:hx:L;
ht = hx/v; %We use the "magic stepsize"
t = 0:ht:tend;

methods = {@upwind,@laxWen};
T = Tcool*ones(length(t),length(x),length(methods));
for i = 1:length(methods)
    meth = methods{i};
    T(:,:,i) = meth(T(:,:,i),hx,ht,a,v,Tcool,Thot);
    figure;
    surf(x,t,T(:,:,i));
    str = func2str(meth);
    title(sprintf('%s - hx = %f',str,hx));

end

end

function T = upwind(T,hx,ht,a,v,Tcool,Thot)
%Solves the problem with the upwind method
[n,m] = size(T);
coeff = v*ht/hx;
for k = 1:n-1
    T(k+1,1) = bcLeft(k*ht,Tcool,Thot);
    T(k+1,2:m) = (1-coeff-a*ht)*T(k,2:m) + coeff*T(k,1:m-1) + a*ht*Tcool;
end
end
```

```
function T = laxWen(T,hx,ht,a,v,Tcool,Thot)
%Solves the problem with the Lax-Wendroff method
[n,m] = size(T);
c1 = ht*v*(1+ht*v/hx-a*ht)/(2*hx);
c2 = 1-a*ht-v*v*ht*ht/(hx*hx)+a*a*ht*ht/2;
c3 = ht*v*(-1+ht*v/hx+a*ht)/(2*hx);
c4 = ht*a*(1-ht*a/2)*Tcool;
for k = 1:n-1
    T(k+1,1) = bcLeft(k*ht,Tcool,Thot);
    T(k+1,2:m-1) = c1*T(k,1:m-2) + c2*T(k,2:m-1) + c3*T(k,3:m) + c4;
    T(k+1,m) = 2*T(k+1,m-1) - T(k+1,m-2);
end
end

function T = bcLeft(t,Tcool,Thot)
%Gives the left boundary condition
if t <= 0.5
    T = Tcool + (Thot-Tcool)*sin(pi*t);
elseif t <= 4
    T = Thot;
else
    T = Thot + Tcool*sin(pi*(t-4));
end
end
```