Exercise set 5

Numerical Methods for Hyperbolic Partial Differential Equations
IMATH, FS-2020
Lecturer: Dr. Philipp Öffner
Teaching Assistant: Davide Torlo

Problem 5.1 Lax–Friedrichs scheme (5pts + 1 extra point)

We study in this exercise the Lax–Friedrichs scheme. For a conservation law

$$\partial_t u(t,x) + \partial_x f(u(t,x)) = 0, \tag{1}$$

with $x \in [a,b]$ and $t \in [0,T]$, the method reads

$$u_j^{n+1} = \frac{u_{j-1}^n + u_{j+1}^n}{2} - \frac{\Delta t}{\Delta x}\frac{f(u_{j+1}^n) - f(u_{j-1}^n)}{2}, \tag{2}$$

with the usual FD notation.

1. Prove that the scheme is first order accurate: define the local truncation error as

$$E = u(t^{n+1}, x_j) - u_j^{n+1}, \tag{3}$$

supposing that $u(t^n, x_j) = u_j^n$ for all j and that $u(t,x)$ is regular enough, prove that $E = \mathcal{O}(\Delta t^2)$, with the usual CFL conditions $\Delta t = \lambda \Delta x$.

Solution

Let us use the Taylor expansion in u_j^n for all the terms in (2) and the properties of the equation (1). Using the notation $u = u_j^n$, we have that

$$\begin{aligned}
u_j^{n+1} =& \frac{u_{j-1}^n + u_{j+1}^n}{2} - \frac{\Delta t}{\Delta x}\frac{f(u_{j+1}^n) - f(u_{j-1}^n)}{2} = \\
=& \frac{u - \Delta x u_x + \frac{\Delta x^2}{2} u_{xx} + u + \Delta x u_x + \frac{\Delta x^2}{2} u_{xx}}{2} \\
& - \frac{\Delta t}{\Delta x}\frac{f(u) + \Delta x f(u)_x + \frac{\Delta x^2}{2} f(u)_{xx} - f(u) + \Delta x f(u)_x - \frac{\Delta x^2}{2} f(u)_{xx}}{2} + \mathcal{O}(\Delta x^3) = \\
=& u + \frac{\Delta x^2}{2} u_{xx} - \Delta t f(u)_x + \mathcal{O}(\Delta x^3).
\end{aligned}$$

Now, we can compute the local truncation error

$$\begin{aligned}
E =& u + \Delta t u_t + \frac{\Delta t^2}{2} u_{tt} - u - \frac{\Delta x^2}{2} u_{xx} + \Delta t f(u)_x + \mathcal{O}(\Delta x^3) = \\
=& \Delta t \underbrace{(u_t + f(u)_x)}_{=0} + \frac{\Delta x^2}{2}(\lambda^2 u_{tt} - u_{xx}) + \mathcal{O}(\Delta x^3) = \mathcal{O}(\Delta t^2)
\end{aligned}$$

2. Find the conditions under which the scheme is von Neumann stable (use periodic boundary conditions and linear transport equation $f(u) = c \cdot u$).

Solution

With the usual, we have that

$$e^{\alpha \Delta t} u_j^n = \frac{e^{i\Delta x k} + e^{-i\Delta x k}}{2} u_j^n - \lambda c \frac{e^{i\Delta x k} - e^{-i\Delta x k}}{2} u_j^n = \tag{4}$$

$$= \left(\cos(\Delta x k) + i\lambda c \sin(\Delta x k)\right) u_j^n. \tag{5}$$

Then we know that $|e^{\alpha \Delta t}|^2 = \cos^2(\Delta x k) + \lambda^2 c^2 \sin^2(\Delta x k)$. The Lax–Friedrichs scheme is von Neumann stable if $\lambda|c| \leq 1$, the usual CFL conditions.

3. Code the method for a linear transport equation with periodic boundary conditions. Pass as input c the speed of the transport equation, N number of subintervals of the domain, CFL number ($\Delta t := \text{CFL}\Delta x/|c|$), T the final time, a and b the domain extrema, u_0 the initial conditions.

4. Test the code with $c = 1$, $N = 200$, $\text{CFL} = 0.9$, $T = 1$, $a = -1$, $b = 1$, $u_0 = \cos(\pi x)$.

5. Check the numerical order of accuracy: for number of cells $N \in \{2^k | k = 1, \dots, 10\}$, run the Lax–Friedrichs scheme and compute the final $\mathbb{L}^2$ error with respect to the exact solution

$$\|u(T) - u_{ex}(T)\|_2 := \sqrt{\Delta x \sum_{j=1}^{N} \left(u(T, x_j) - u_{ex}(T, x_j)\right)^2}. \tag{6}$$

Plot the error vs Δx and a reference first order decay. Verify that they have the same rate of convergence.

Solution

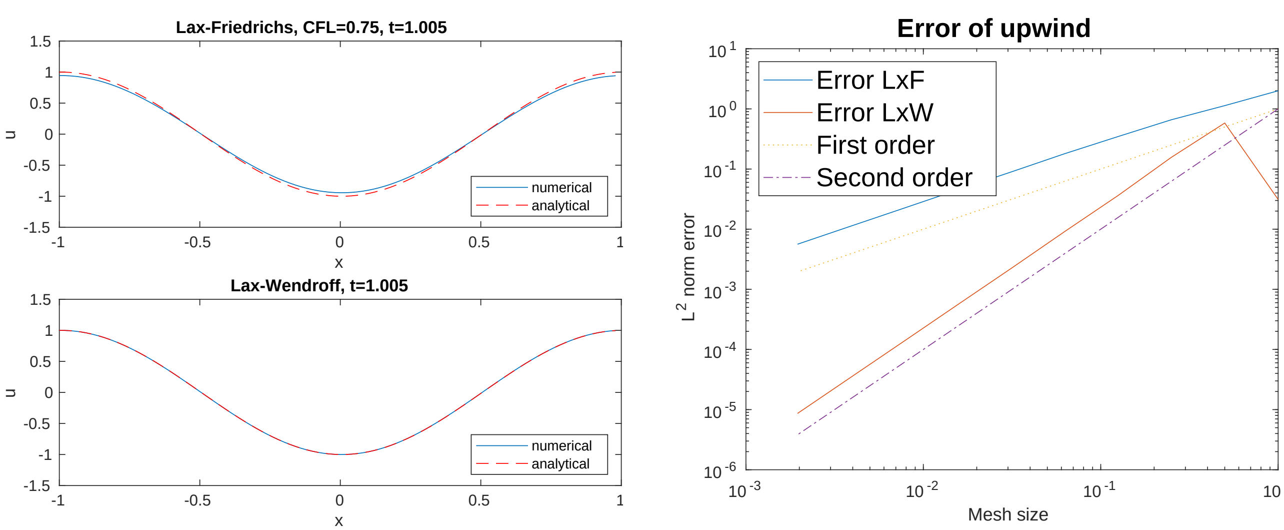

Figure 1: Lax Friedrichs and Lax Wendroff schemes CFL=0.75: test on left, error on right

6. (Extra 1 point) Run again the script for the order convergence with CFL=1. What do you observe and why?

Solution

If CFL are 1, i.e., $\lambda|c| = 1$, we have that in the local truncation error, the second order term

$$\frac{\Delta x^2}{2}\left(\lambda^2 u_{tt} - u_{xx}\right) = \left(\lambda^2 c^2 u_{xx} - u_{xx}\right) = 0. \tag{7}$$

We showed like this that the scheme becomes at least a second order scheme. One can proceed with higher order terms and see that all of them vanish, resulting in an exact scheme. More precisely, the even terms of the spatial derivatives come from the term $\frac{u^n_{j-1}+u^n_{j+1}}{2}$, while the odd terms come from the $\Delta t \frac{f(u^n_{j+1})-f(u^n_{j-1})}{\Delta x}$.

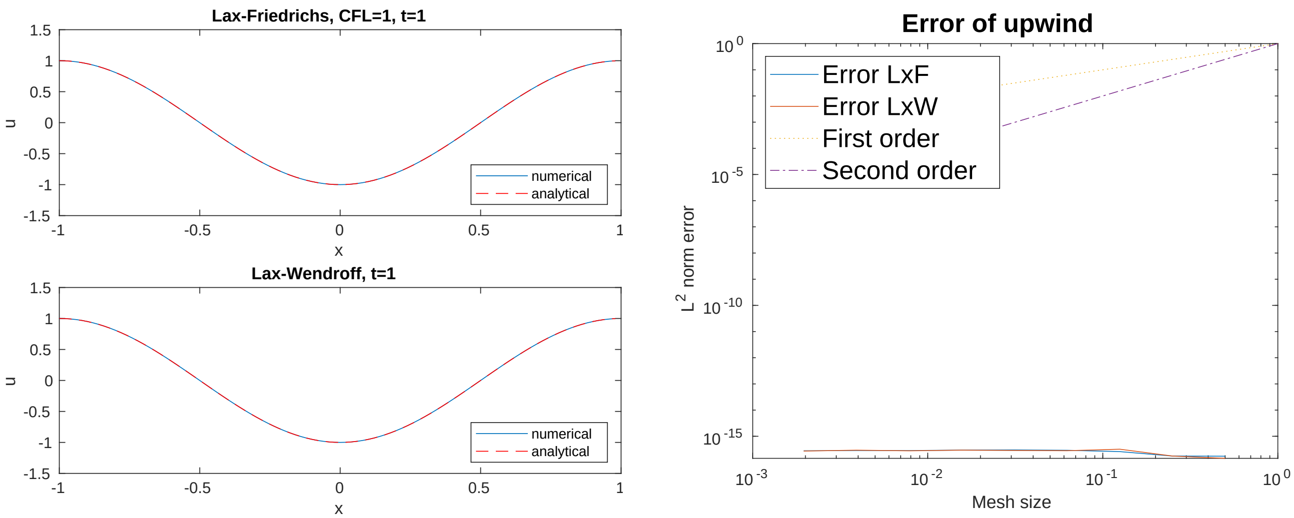

Figure 2: Lax Friedrichs and Lax Wendroff schemes CFL=1: test on left, error on right

```
function [u,t,x, varargout] = lxf(c, N, cfl, T, a,b, u0, varargin)
%LAXFR   Solve the advection equation u_t + c u_x = 0 using the
%   upwind/downwind scheme. The solution is computed on the periodic domain
%   [-a, b) up to t=tMax.

    if nargin>7
        with_error=1;
    else
        with_error=0;
    end

  % Create the mesh
  x = linspace(a, b, N+1);
  dx = x(2)-x(1);
  x = [x(N), x];      % Mesh with ghost cells {x_0, ..., x_{N+1}}
  j = 2 : N+1;     % Index into the internal mesh  1,...,N

  % Set initial data and advection speed
  aMax = abs(c);
    if with_error
        u_exact=@(x,t) u0(x-c*t);
      uu_exact(1,:) = u_exact(x,0);
    end
    u(1,:) = u0(x);


  % Set the timestep and make sure that tMax/dt is an integer
  dt = cfl * dx/aMax;
  nt = ceil(T / dt)+1;
  t = [0:nt-1]*dt; % linspace(0, T, nt);
  %dt = t(2)-t(1);
  lambda = dt/dx;

  % Run the simulation
```

```
  for n = 2:nt
    % Update the solution
    u(n,j) = (u(n-1,j+1)+u(n-1,j-1))/2 - lambda/2*c*(u(n-1,j+1)-u(n-1,j-1));

    % Set boundary conditions
    u(n,1) = u(n,N+1);
    u(n,N+2) = u(n,2);
        if with_error
            uu_exact(n,:) = u_exact(x,t(n));
        end
    end
    if with_error
      errL2end= norm(u(end,:)-uu_exact(end,:))*sqrt(dx);
        varargout={uu_exact,errL2end};
    else
        varargout={};
    end
  % Remove the ghost cell to the left
  x = x(2:end-1);
  u = u(:,2:end-1);
end
```

Listing 1: lxf.m

```
function [u,t,x, varargout] = lxw(c, N, cfl, T, a,b, u0, varargin)
%LAXFR   Solve the advection equation u_t + c u_x = 0 using the
%   upwind/downwind scheme. The solution is computed on the periodic domain
%   [-a, b) up to t=tMax.

    if nargin>7
        with_error=1;
    else
        with_error=0;
    end

  % Create the mesh
  x = linspace(a, b, N+1);
  dx = x(2)-x(1);
  x = [x(N), x];    % Mesh with ghost cells {x_0, ..., x_{N+1}}
  j = 2 : N+1;    % Index into the internal mesh  1,...,N

  % Set initial data and advection speed
  aMax = abs(c);
    if with_error
        u_exact=@(x,t) u0(x-c*t);
      uu_exact(1,:) = u_exact(x,0);
    end
    u(1,:) = u0(x);


  % Set the timestep and make sure that tMax/dt is an integer
  dt = cfl * dx/aMax;
  nt = ceil(T / dt)+1;
  t = [0:nt-1]*dt; % linspace(0, T, nt);
  %dt = t(2)-t(1);
  lambda = dt/dx;

  % Run the simulation
  for n = 2:nt
    % Update the solution
    u(n,j) = u(n-1,j)- lambda/2*c*(u(n-1,j+1)-u(n-1,j-1)) ...
                + c.^2*lambda^2/2*(u(n-1,j-1)-2*u(n-1,j)+u(n-1,j+1));

    % Set boundary conditions
```

```
    u(n,1) = u(n,N+1);
    u(n,N+2) = u(n,2);
        if with_error
            uu_exact(n,:) = u_exact(x,t(n));
        end
    end
    if with_error
      errL2end= norm(u(end,:)-uu_exact(end,:))*sqrt(dx);
        varargout={uu_exact,errL2end};
    else
        varargout={};
    end
  % Remove the ghost cell to the left
  x = x(2:end-1);
  u = u(:,2:end-1);
end
```

Listing 2: lxw.m

```
% test lxf

c=1;
%cfl=0.75;
T=1;
a=-1;
b=1;
u0=@(x) cos(pi*x);

%% run lxf scheme
%
Nx = 100;
tfin = 1.;
cfl = 1;
with_error=1;
plot_evolution=1;
%
[ulxf,t,x,ex,err] = lxf(c, Nx, cfl, tfin, a,b, u0, with_error);
[ulxw,t,x,ex,err] = lxw(c, Nx, cfl, tfin, a,b, u0, with_error);

%% plot evolution
dx = x(2)-x(1);
u0 = @(x) cos(pi*x);
fig=figure(1);
if(plot_evolution)
    for n=[1:ceil(numel(t)/20):numel(t),numel(t)]
        %
        figure(1)
        subplot(211)
        plot(x,ulxf(n,:),...
            x,u0(x-c*t(n)),'r--')
        legend('numerical','analytical','Location','SE')
        xlabel x
        ylabel u
        title(sprintf('Lax-Friedrichs, CFL=%g, t=%g',cfl,t(n)))
        ylim([-1.5,1.5])

        subplot(212)
        plot(x,ulxw(n,:),...
            x,u0(x-c*t(n)),'r--')
        legend('numerical','analytical','Location','SE')
        xlabel x
        ylabel u
        title(sprintf('Lax-Wendroff, t=%g',t(n)))
        ylim([-1.5,1.5])
```

```
        %
        drawnow
        pause(.02)
    end
end

saveas(fig, 'LxFtest.pdf')

%% plot TV instability for upwind

fig=figure
TVlxf=zeros(size(t));
ulxf_1=zeros(size(ulxf));
ulxf_1(:,1:end-1)=ulxf(:,2:end);
ulxf_1(:,end)=ulxf(:,1);

TVlxw=zeros(size(t));
ulxw_1=zeros(size(ulxw));
ulxw_1(:,1:end-1)=ulxw(:,2:end);
ulxw_1(:,end)=ulxw(:,1);
for n=1:numel(t)
    TVlxf(n)= sum(abs(ulxf(n,:)-ulxf_1(n,:)))*dx;
    TVlxw(n)= sum(abs(ulxw(n,:)-ulxw_1(n,:)))*dx;
    %
end

semilogy(t,TVlxf,'DisplayName','LxF')
hold on
semilogy(t,TVlxw,'DisplayName','LxW')

xlabel('t','FontSize',16)
ylabel('TotalVariation','FontSize',16)
title('Total Variation','FontSize',16)
legend('FontSize',15,'Location','NW')
hold off
saveas(fig,'TVlxflxw.pdf')

%% plot error convergence
tfin = 1.;
cfl =0.75;

with_error=1;
nn=10;
Nxs=2.^[1:nn];
errlxf=zeros(size(Nxs));
errlxw=zeros(size(Nxs));
for k=1:nn
    Nx=Nxs(k);
    hs(k)=2/Nx;
    [ulxf,t,x,ex, err] = lxf(c, Nx, cfl, tfin, a,b, u0, 1 );
    errlxf(k)=err;
    [ulxw,t,x,ex, err] = lxw(c, Nx, cfl, tfin, a,b, u0, 1 );
    errlxw(k)=err;
end

fig=figure()
loglog(hs,errlxf,'DisplayName', 'Error LxF')
hold on
loglog(hs,errlxw,'DisplayName', 'Error LxW')
title('Error of upwind','FontSize',16)
ylabel('L^2 norm error')
xlabel('Mesh size')
hold on
```

```
loglog(hs,hs,':','DisplayName','First order')
loglog(hs,hs.^2,'-.','DisplayName','Second order')
legend('Location','NW','FontSize',16)
saveas(fig,sprintf('errorLxFcfl%g.pdf',cfl))
```

Listing 3: testLxF.m

Problem 5.2 Source terms (5pts + 2 extra pts)

Consider now the linear transport equation with a source term

$$\partial_t u(t,x) + c\partial_x u(t,x) = S(u(x,t)) \tag{8}$$

with initial condition $u_0(x)$.

1. Compute the exact solution of (8) for smooth initial data u_0 when the source is $S(u(t,x)) = s \cdot u(t,x)$ for $s \in \mathbb{R}$.

   Solution

   We know the solution of the homogeneous problem $u_t + cu_x = 0$ which is $u(t,x) = u_0(x-ct)$. To obtain the solution of the source problem, we can consider an ODE, with initial condition the solution just found, i.e.

   $$\begin{cases} \frac{du}{dt} = su \\ u_0 = u_0(x-ct). \end{cases} \tag{9}$$

   The solution is clearly $u(x,t) = e^{st}u_0(x-ct)$ and we can verify it substituting the solution in the equation. We get that

   $$\partial_t u(t,x) + c\partial_x u(t,x) = S(u(x,t)) \tag{10}$$
   $$se^{st}u_0(x-ct) - ce^{st}u_0'(x-ct) + ce^{st}u_0'(x-ct) = se^{st}u_0(x-ct). \tag{11}$$

2. Write an explicit first order scheme based on the Lax–Friedrichs (2) to solve the linear equation with a source term. The update formula for u_j^{n+1} should depend only on u_{j-1}^n, u_j^n and u_{j+1}^n, i.e., the *footprint* of the stencil is 3 (there are infinitely many possibilities). Prove that it is first order accurate with Taylor expansions as in (3).

   Solution

   There are many options as we need just a first order scheme. One non totally trivial is

   $$u_j^{n+1} = \frac{u_{j-1}^n + u_{j+1}^n}{2} - \frac{\Delta t}{\Delta x}\frac{f(u_{j+1}^n) - f(u_{j-1}^n)}{2} + \Delta t \frac{S(u_{j-1}^n) + S(u_{j+1}^n)}{2}. \tag{12}$$

   We can actually write all the possible schemes as

   $$u_j^{n+1} = \frac{u_{j-1}^n + u_{j+1}^n}{2} - \frac{\Delta t}{\Delta x}\frac{f(u_{j+1}^n) - f(u_{j-1}^n)}{2} + z_{-1}S(u_{j-1}^n) + z_0 S(u_j^n) + z_1 S(u_{j+1}^n). \tag{13}$$

Then we can check which condition we need to impose on the coefficients zs in order to get a first order scheme. Consider now the local truncation error

$$\begin{aligned}
E =&u + \Delta t u_t + \frac{\Delta t^2}{2} u_{tt} - u - \frac{\Delta x^2}{2} u_{xx} + \Delta t f(u)_x - \left(z_{-1} S(u^n_{j-1}) + z_0 S(u^n_j) + z_1 S(u^n_{j+1}) \right) + \mathcal{O}(\Delta x^3) = \\
=&\Delta t u_t + \frac{\Delta t^2}{2} u_{tt} - \frac{\Delta x^2}{2} u_{xx} + \Delta t f(u)_x - (z_{-1} + z_0 + z_1)\, S(u) + \Delta t \Delta x\, (z_1 - z_{-1})\, S(u)_x + \mathcal{O}(\Delta x^3) = \\
=&\Delta t \left(u_t + f(u)_x - \frac{z_{-1} + z_0 + z_1}{\Delta t} S(u) \right) + \frac{\Delta t^2}{2} u_{tt} - \frac{\Delta x^2}{2} u_{xx} + \Delta t \Delta x\, (z_1 - z_{-1})\, S(u)_x + \mathcal{O}(\Delta x^3).
\end{aligned}$$

Since we want a 1st order scheme, we just have to check that the $\mathcal{O}(\Delta x^2)$ should vanish. Knowing that $u_t + f(u)_x - S(u) = 0$, it is clear that $z_{-1} + z_0 + z_1 = \Delta t$. This condition guarantees that the scheme is consistent and first order accurate.

3. Create a new function that implements your method for the linear equation with the linear source term $S(u) = s \cdot u$ (modify the function of the previous exercise, adding also an extra input s).
4. Test it with $s = -0.5$ and $c = 1$, CFL $= 0.9$, $T = 1$, $a = -1$, $b = 1$, $u_0 = \cos(\pi x)$ and compute the numerical error decay as in (6). Is the accuracy of the scheme correct?
5. Consider the following semi–implicit scheme based on the Lax–Wendroff method

$$u^{n+1}_j = u^n_j - \frac{c\Delta t}{2\Delta x}(u^n_{j+1} - u^n_{j-1}) + \frac{c^2 \Delta t^2}{2\Delta x^2}\left(u^n_{j+1} - 2u^n_j + u^n_{j-1}\right) + \Delta t \frac{S(u^{n+1}_j) + S(u^n_j)}{2} - \frac{c\Delta t^2}{4\Delta x}\left(S(u^n_{j+1}) - S(u^n_{j-1})\right). \tag{14}$$

What is its order of accuracy? Use the Taylor expansion.

Solution

Using the notation $u = u^n_j$, $\lambda = \frac{\Delta t}{\Delta x}$, we expand with the Taylor series all the terms of the local truncation error till the second order terms:

$$\begin{aligned}
-E =& -u - \Delta t u_t - \frac{\Delta t^2}{2} u_{tt} + u - c \underbrace{\lambda \Delta x}_{=\Delta t} u_x + c^2 \underbrace{\lambda^2 \frac{\Delta x^2}{2}}_{=\Delta t^2/2} u_{xx} + \Delta t S(u) + \frac{\Delta t^2}{2} S(u)_t - \frac{c\Delta t^2}{2} S(u)_x + \mathcal{O}(\Delta t^3) = \\
=& -\Delta t \underbrace{(u_t + c u_x - S(u))}_{=0} - \frac{\Delta t^2}{2}\left(u_{tt} - c^2 u_{xx} - S(u)_t + c S(u)_x\right) + \mathcal{O}(\Delta t^3) = \\
=& -\frac{\Delta t^2}{2}\left((-c u_x + S(u))_t - c^2 u_{xx} - S(u)_t + c S(u)_x\right) + \mathcal{O}(\Delta t^3) = \\
=& -\frac{\Delta t^2}{2}\left((-c(-c u_x + S(u))_x) + S'(u) u_t - c^2 u_{xx} - S(u)_t + c S(u)_x\right) + \mathcal{O}(\Delta t^3) = \\
=& -\frac{\Delta t^2}{2}\left(c^2 u_{xx} - c S'(u) u_x + S'(u) u_t - c^2 u_{xx} - S(u)_t + c S(u)_x\right) + \mathcal{O}(\Delta t^3) = \mathcal{O}(\Delta t^3).
\end{aligned}$$

6. (Extra 2 points) Implement the method (14) for the linear source $S(u) = s \cdot u$ and periodic boundary conditions and test the numerical accuracy of the scheme.

Solution

```
function [u,t,x, varargout] = lxfSource(c, N, cfl, T, a,b, u0,s, varargin)
%LAXFR   Solve the advection equation u_t + c u_x = 0 using the
%   upwind/downwind scheme. The solution is computed on the periodic domain
%   [-a, b) up to t=tMax.

```

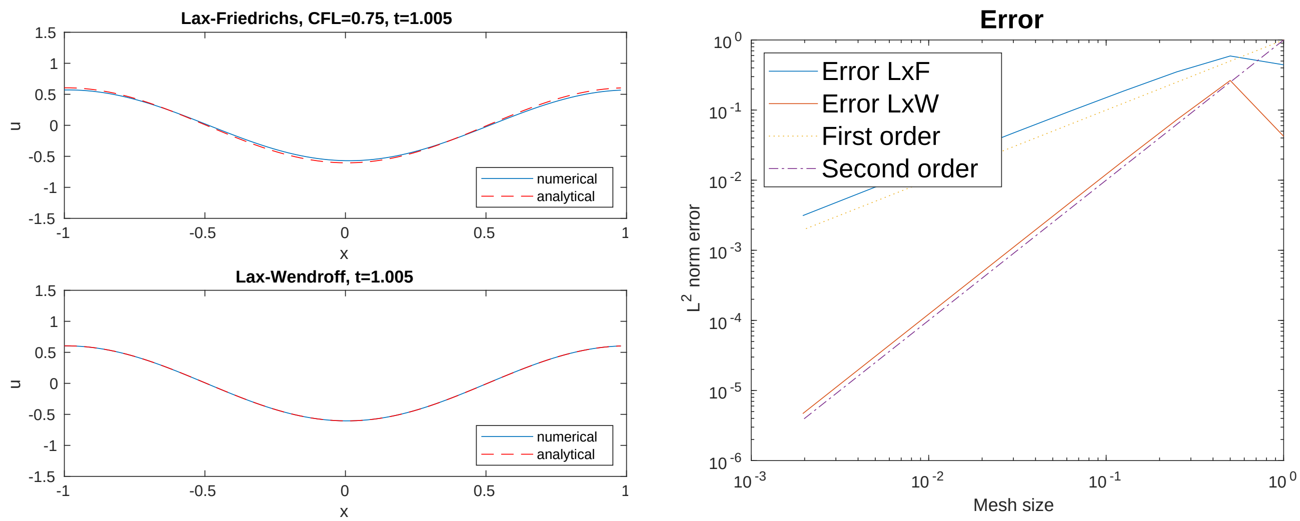

Figure 3: Lax Friedrichs with central source and Lax Wendroff type scheme (14) at CFL=0.8: test on left, error on right

```
    if nargin>8
        with_error=1;
    else
        with_error=0;
    end

  % Create the mesh
  x = linspace(a, b, N+1);
  dx = x(2)-x(1);
  x = [x(N), x];     % Mesh with ghost cells {x_0, ..., x_{N+1}}
  j = 2 : N+1;    % Index into the internal mesh  1,...,N

  % Set initial data and advection speed
  aMax = abs(c);
    if with_error
        u_exact=@(x,t) exp(s*t)*u0(x-c*t);
      uu_exact(1,:) = u_exact(x,0);
    end
    u(1,:) = u0(x);


  % Set the timestep and make sure that tMax/dt is an integer
  dt = cfl * dx/aMax;
  nt = ceil(T / dt)+1;
  t = [0:nt-1]*dt; % linspace(0, T, nt);
  %dt = t(2)-t(1);
  lambda = dt/dx;

  % Run the simulation
  for n = 2:nt
    % Update the solution
    u(n,j) = (u(n-1,j+1)+u(n-1,j-1))/2 - lambda/2*c*(u(n-1,j+1)-u(n-1,j-1)) ...
                +dt*0.5*s*(u(n-1,j+1)+u(n-1,j-1));

    % Set boundary conditions
    u(n,1) = u(n,N+1);
    u(n,N+2) = u(n,2);
        if with_error
            uu_exact(n,:) = u_exact(x,t(n));
```

```
        end
    end
    if with_error
      errL2end= norm(u(end,:)-uu_exact(end,:))*sqrt(dx);
        varargout={uu_exact,errL2end};
    else
        varargout={};
    end
  % Remove the ghost cell to the left
  x = x(2:end-1);
  u = u(:,2:end-1);
end
```

Listing 4: lxfSource.m

```
function [u,t,x, varargout] = lxwSourceImplicit(c, N, cfl, T, a,b, u0, s, varargin)
%LAXFR   Solve the advection equation u_t + c u_x = s u using the
%   upwind/downwind scheme. The solution is computed on the periodic domain
%   [-a, b) up to t=tMax.

    if nargin>8
        with_error=1;
    else
        with_error=0;
    end

  % Create the mesh
  x = linspace(a, b, N+1);
  dx = x(2)-x(1);
  x = [x(N), x];    % Mesh with ghost cells {x_0, ..., x_{N+1}}
  j = 2 : N+1;    % Index into the internal mesh  1,...,N

  % Set initial data and advection speed
  aMax = abs(c);
    if with_error
        u_exact=@(x,t) exp(s*t)*u0(x-c*t);
      uu_exact(1,:) = u_exact(x,0);
    end
    u(1,:) = u0(x);


  % Set the timestep and make sure that tMax/dt is an integer
  dt = cfl * dx/aMax;
  nt = ceil(T / dt)+1;
  t = [0:nt-1]*dt; % linspace(0, T, nt);
  %dt = t(2)-t(1);
  lambda = dt/dx;

  % Run the simulation
  for n = 2:nt
    % Update the solution
    u(n,j) = u(n-1,j)- lambda/2*c*(u(n-1,j+1)-u(n-1,j-1)) ...
            + c.^2*lambda^2/2*(u(n-1,j-1)-2*u(n-1,j)+u(n-1,j+1))...
            +dt*0.5*s*(u(n-1,j)) ...
            -c*dt^2/4/dx*s*(u(n-1,j+1)-u(n-1,j-1));
    u(n,j)=u(n,j)/(1-dt*s/2);
    % Set boundary conditions
    u(n,1) = u(n,N+1);
    u(n,N+2) = u(n,2);
        if with_error
            uu_exact(n,:) = u_exact(x,t(n));
        end
    end
    if with_error
```

```
      errL2end= norm(u(end,:)-uu_exact(end,:))*sqrt(dx);
        varargout={uu_exact,errL2end};
     else
        varargout={};
     end
  % Remove the ghost cell to the left
  x = x(2:end-1);
  u = u(:,2:end-1);
end
```

Listing 5: lxwSourceImplicit.m

```
% test lxf

clear all
close all

c=1;
cfl=0.75;
tfin=1;
a=-1;
b=1;
u0=@(x) cos(pi*x);

%% run lxf scheme
%
Nx = 100;
with_error=1;
plot_evolution=1;
s=-0.5;
%
[ulxf,t,x,ex,err] = lxfSource(c, Nx, cfl, tfin, a,b, u0, s, with_error);
[ulxw,t,x,ex,err] = lxwSourceImplicit(c, Nx, cfl, tfin, a,b, u0, s, with_error);

%% plot evolution
dx = x(2)-x(1);
u0 = @(x) cos(pi*x);
fig=figure(1)
if(plot_evolution)
    for n=[1:ceil(numel(t)/20):numel(t),numel(t)]
        %
        figure(1)
        subplot(211)
        plot(x,ulxf(n,:),...
            x,exp(s*t(n))*u0(x-c*t(n)),'r--')
        legend('numerical','analytical','Location','SE')
        xlabel x
        ylabel u
        title(sprintf('Lax-Friedrichs, CFL=%g, t=%g',cfl,t(n)))
        ylim([-1.5,1.5])

        subplot(212)
        plot(x,ulxw(n,:),...
            x,exp(s*t(n))*u0(x-c*t(n)),'r--')
        legend('numerical','analytical','Location','SE')
        xlabel x
        ylabel u
        title(sprintf('Lax-Wendroff, t=%g',t(n)))
        ylim([-1.5,1.5])

        %
        drawnow
        pause(.02)
    end
```

```
end


saveas(fig, 'LxFSorucetest.pdf')

%% plot error convergence
tfin = 1.;
cfl =0.8;

with_error=1;
nn=10;
Nxs=2.^[1:nn];
errlxf=zeros(size(Nxs));
errlxw=zeros(size(Nxs));
for k=1:nn
    Nx=Nxs(k);
    hs(k)=2/Nx;
    [ulxf,t,x,ex, err] = lxfSource(c, Nx, cfl, tfin, a,b, u0, s, 1 );
    errlxf(k)=err;
    [ulxw,t,x,ex, err] = lxwSourceImplicit(c, Nx, cfl, tfin, a,b, u0, s, 1 );
    errlxw(k)=err;
end

fig=figure()
loglog(hs,errlxf,'DisplayName', 'Error LxF')
hold on
loglog(hs,errlxw,'DisplayName', 'Error LxW')
title('Error','FontSize',16)
ylabel('L^2 norm error')
xlabel('Mesh size')
hold on
loglog(hs,hs,':','DisplayName','First order')
loglog(hs,hs.^2,'-.','DisplayName','Second order')
legend('Location','NW','FontSize',16)
saveas(fig,'errorLxFLxWSource.pdf')
```

Listing 6: testLxFSource.m

Submit the code for both exercises.

Organiser: Davide Torlo, Office: home (davide.torlo@math.uzh.ch)

Published: Apr 2, 2020

Due date: Apr 9, 2020, h10.00 (use the upload tool of my.math.uzh.ch, see wiki.math.uzh.ch/public/student_upload_homework or if you have troubles send me an email).