Exercise Set 4: The Finite Difference Method

Exercise 1 (Finite Differences Method)
Define the following concepts:

(a) Consistency;

(b) Stability; and,

(c) Convergence.

Discuss the importance of these concepts, how they relate to each other, and how in general they can be established.

Solution 1

1. *Consistency.* A method is consistent if its local truncation error T_k satisfies

$$\tau_k(x,t) = O\left(k^p\right) + O\left(h^q\right) \qquad \text{where} \qquad p, q > 0 \ . \tag{1}$$

   Essentially, consistency tells us that we are approximating the solution of the correct PDE.

2. *Stability.* A method $v^{n+1} = \mathcal{H}_k v^n$ is stable if for each $T > 0$ there exist constants C and k_0, such that

$$\|\mathcal{H}_k^n\| \leq C \qquad\qquad 0 \leq nk \leq T \ , \quad 0 < k < k_0 \ . \tag{2}$$

3. *Convergence.* A method is convergent if the error E_k satisfies

$$\lim_{k \to 0} \max_{0 \leq kn \leq T} \|E_k\left(\cdot, kn\right)\| = 0 \ . \tag{3}$$

Exercise 2 (Leapfrog Method)
The forward time, center space finite difference approximation to the advection equation $u_t + au_x = 0$ leads to a method which is *unstable*. So let us try something different. By approximating the time derivative by a centered difference, instead of the forward Euler discretization, we get the Leapfrog method,

$$v_j^{n+1} = v_j^{n-1} - \frac{ak}{h}(v_{j+1}^n - v_{j-1}^n) \ . \tag{4}$$

(i) Draw the stencil of (4).

(ii) Show that (4) is second order accurate in both space and time.

(iii) What is an obvious disadvantage of the Leapfrog method compared to the Lax-Friedrichs or Lax-Wendroff methods?

Solution 2

1. Let u be a smooth solution of $u_t + au_x = 0$. The local truncation error for the Leapfrog scheme is given by

$$k\tau_k(x,t) = u\left(x, t+k\right) - u\left(x, t-k\right) + \frac{ak}{h}\left(u\left(x+h, t\right) - u\left(x-h, t\right)\right) \ . \tag{5}$$

   Next, we expand all the terms on the right hand side of (5) about (x,t). For example, we have $u\left(x, t+k\right)$ and $u\left(x, t-k\right)$ given by

$$u\left(x, t+k\right) = u + u_t k + \frac{1}{2} u_{tt} k^2 + O\left(k^3\right) \tag{6}$$

and

$$u\left(x,t-k\right)=u-u_tk+\frac{1}{2}u_{tt}k^2+O\left(k^3\right)\ , \tag{7}$$

where, for simplicity of notation, u, u_t and u_{tt} stands for $u(x,t)$, $u_t(x,t)$ and $u_{tt}(x,t)$, respectively. It is, thus, clear that

$$u\left(x,t+k\right)-u\left(x,t-k\right)=2ku_t+O\left(k^3\right)\ . \tag{8}$$

By repeating the calculation also for the translations in space $u(x\pm h,t)$, we get

$$k\tau_k(x,t)=2ku_t+O\left(k^3\right)+\frac{ak}{h}\left(2hu_x+O\left(h^3\right)\right)\ , \tag{9}$$

which implies

$$\tau_k(x,t)=2\left(u_t+au_x\right)+O\left(k^2\right)+O\left(h^2\right)\ . \tag{10}$$

Since u satisfies $u_t+au_x=0$, the local truncation error is simply $\tau_k(x,t)=O\left(k^2\right)+O\left(h^2\right)$.

2. Notice that at the n-th time step, to calculate v^{n+1}, the Leapfrog scheme requires the values, not only v^n, but also of v^{n-1}. Compared to the LF and LW schemes which use only v^n to calculate v^{n+1}, this is a clear disadvantage. In computations, keeping the numerical solution at more than one time level multiplies the size of memory required for the program.

   Another issue is obtaining the first values. In our example, the values of v^0 are obtained directly from the initial data, however the Leapfrog scheme requires another time level to work. So we must use other means to get v^1, and only then can we use the Leapfrog method to advance.

Exercise 3 (Stability of the Lax-Friedrichs Method)
As a possible numerical method for the linear transport equation $u_t+au_x=0$, consider the Lax-Friedrichs method

$$v_j^{n+1}=\frac{1}{2}(v_{j+1}^n+v_{j-1}^n)-\frac{ak}{2h}(v_{j+1}^n-v_{j-1}^n) \tag{11}$$

Show that this method is stable in the l^∞ norm, provided that k and h satisfy the *CFL condition*

$$\frac{|a|k}{h}\leq 1. \tag{12}$$

Solution 3
To show that the Lax-Friedrichs (LF) scheme is stable provided

$$\frac{|a|k}{h}\leq 1\ , \tag{13}$$

we show that $\|\mathcal{H}^n\|_\infty$ is bounded for all n, where $\mathcal{H}$ is the operator defined by

$$\mathcal{H}v_j=\frac{1}{2}\left(v_{j+1}+v_{j-1}\right)-\frac{ak}{2h}\left(v_{j+1}-v_{j-1}\right)\ . \tag{14}$$

(That is, the LF scheme is given by $v^{n+1}=\mathcal{H}v^n$.) To do so, we suppose v is some grid function, and calculate the norm of $\mathcal{H}v$. We have

$$\left\|\mathcal{H}v^n\right\|_\infty=\left\|\frac{1}{2}(v_{j+1}+v_{j-1})-\frac{ak}{2h}(v_{j+1}-v_{j-1})\right\|_\infty\ , \tag{15}$$

$$\leq\frac{1}{2}\left(|1-\frac{ak}{h}|+|1+\frac{ak}{h}|\right)\left\|v^n\right\|_\infty\ , \tag{16}$$

$$\leq\left\|v^n\right\|_\infty\ , \tag{17}$$

whenever (13) holds. Thus,

$$\|\mathcal{H}^n\|_\infty\leq\|\mathcal{H}\|_\infty^n\leq 1\qquad\qquad\forall n\in\mathbb{N}\ . \tag{18}$$

Exercise 4 (Unconditionally Stable Method)
We have seen in the previous exercise that the Lax-Friedrichs method is stable provided k and h satisfy a CFL condition. A method which is stable for any k and h is said to be *unconditionally stable*. For $a > 0$, prove that the following backward-time backward-space method

$$v_j^{n+1} = v_j^n - \frac{ak}{h}(v_j^{n+1} - v_{j-1}^{n+1})$$

is unconditionally stable in the l^∞ norm.

Solution 4
Let us consider the scheme

$$v_j^{n+1} = v_j^n - \lambda\left(v_j^{n+1} - v_{j-1}^{n+1}\right), \quad \lambda = \frac{ak}{h}. \tag{19}$$

Before trying to show stability, we first re-write the scheme (assuming periodic boundaries) as

$$(1+\lambda)\, v_j^{n+1} - \lambda v_{j-1}^{n+1} = v_j^n \quad \Longrightarrow \quad Av^{n+1} = v_n, \tag{20}$$

where

$$A = \begin{bmatrix} 1+\lambda & 0 & \dots & 0 & -\lambda \\ -\lambda & 1+\lambda & 0 & \dots & 0 \\ \ddots & \ddots & \ddots & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & -\lambda & 1+\lambda \end{bmatrix} \tag{21}$$

Note that we can now write the scheme in the form $v^{n+1} = \mathcal{H}v^n$, where $\mathcal{H} = A^{-1}$, provided A is non-singular. Since A is strictly diagonally dominant, its eigenvalues are non-zero by Gershgorin's theorem. This proves that A is invertible. Furthermore, there is a nice theorem (see the book by Varah below) about a diagonally dominant matrix A, which says

$$\|A^{-1}\|_\infty \leq \frac{1}{\alpha}, \quad \alpha = \min_k \left(|A_{kk}| - \sum_{j \neq k} |A_{kj}| \right). \tag{22}$$

In our case, $\alpha = 1$. Thus, $\|\mathcal{H}\|_\infty = \left\|A^{-1}\right\|_\infty \leq 1$. Since this condition holds independent of any CFL restrictions on h and k, the scheme is unconditionally stable.

References

[1] A lower bound for the smallest singular value of a matrix, by J. M. Varah. *Linear Algebra and its Applications, vol. 11, issue 1, pg 3-5, 1975.*

Exercise 5 (Matlab Implementation)
This exercise involves some programming. Consider the scalar advection equation $u_t + au_x = 0$ with $a = 1$. Let u be the solution of $u_t + au_x = 0$ in $(-1, 1)$ that satisfies initial condition

$$u(x, 0) = u_0(x), \tag{23}$$

where

$$u_0(x) = \begin{cases} 1 & x < 0 \\ 0 & x > 0 \end{cases}, \tag{24}$$

and boundary conditions

$$u(-1, t) = 1 \qquad u(1, t) = 0 \tag{25}$$

We already know that the exact solution for $0 < t < 1$ is given by $u_0(x - at)$, but how well do numerical methods approximate such a problem? In the following consider the schemes

$$\begin{aligned}
\text{Upwind:} \quad & v_j^{n+1} = v_j^n - \frac{ak}{h}(v_j^n - v_{j-1}^n) \\
\text{Lax-Friedrichs:} \quad & v_j^{n+1} = \frac{1}{2}(v_{j+1}^n + v_{j-1}^n) - \frac{ak}{2h}(v_{j+1}^n - v_{j-1}^n) \\
\text{Lax-Wendroff:} \quad & v_j^{n+1} = v_j^n - \frac{ak}{2h}(v_{j+1}^n - v_{j-1}^n) + \frac{(ak)^2}{2h^2}(v_{j+1}^n - 2v_j^n + v_{j-1}^n) \\
\text{Beam-Warming:} \quad & v_j^{n+1} = v_j^n - \frac{ak}{2h}(3v_j^n - 4v_{j-1}^n + v_{j-2}^n) + \frac{(ak)^2}{2h^2}(v_j^n - 2v_{j-1}^n + v_{j-2}^n) \ .
\end{aligned}$$

For each of the schemes above:

1. Implement the scheme in Matlab to solve $u_t + au_x = 0$, in $(-1, 1)$, with (24), and (25) in the time interval $t \in [0, 0.5]$. For your computations use $h = 0.0025$ and $k/h = 0.5$.
2. Visualize the numerical solution and the exact solution.
3. Comment qualitatively on the solutions' behavior.
4. Write a script to generate a log-log plot of the error as a function of the resolution (either h, or the number of points in the spatial grid), while keeping the ratio $k/h = 0.5$ fixed (why is this important?).
5. Use the plot to deduce the accuracy of the method.
6. Compare your results with the accuracy you would expect on a smooth solution.

Solution 5 1. Matlab code for implementing the schemes to solve the advection equation can be found on the last two pages of this solution manual.

2. In Figure 1, $u(x, 0.5)$ is plotted as solved by the Upwind, Lax-Friedrichs, Lax-Wendroff and Beam-Warming methods.
3. Both the upwind and Lax-Friedrichs schemes capture the discontinuity. The upwind scheme, however, seems to produce less numerical dissipation. This is because the upwind scheme exploits that fact that information travels only in one direction. The higher order methods Lax-Wendroff and Beam-Warming both introduce oscillations around the discontinuities.

(d),(e) See Figure 2.

4. Notice how, on this problem with non-smooth solutions, the rate of convergence of the first order methods is now $O\left(h^{1/2}\right)$, while the rate of convergence of the second order methods seem to be $O\left(h^{0.6}\right)$ at best.

```
% Solution04: Problem 5
% This script was written for EPFL MATH459, Numerical Methods for
% Conservation Laws. The scalar advection equation du/dx + du/dt = 0
% with riemann initial data: u = 1 if x<0,  0 if x>0.
% The problem is solved using the following schemes:
%       1. Upwind
%       2. Lax-Friedrichs
%       3. Lax-Wendroff
%       4. Beam-Warming
% The solution is visualized and the accuracy of the scheme tested.

clc
clear all
close all

% Scheme Options: UW --> Upwind
%                 LF --> Lax-Friedrichs
%                 LW --> Lax-Wendroff
%                 BW --> Beam-Warming
% Task Options  : Solve --> To simply run the scheme with h=0.0025 and k/h
   = 0.5
%                 Acc   --> Test accuracy of the scheme by generating Log-
   Log
%                           plots
Scheme      = 'BW';
Task        = 'Acc';

% Advection speed and final time
a  = 1;
Tf = 0.5;

% Initial condition
U0 =@(x,t) 1*(x-a*t < 0);

% Set the function H for the scheme
%   u^{n+1} = H(u^n)
% as well as the number of ghost cells Ng needed at each end of the mesh (
   for
% boundary conditions) and the scheme name for saving solutions.
% Here lam = a*k/h and U is the extended solution vector
if(strcmp(Scheme,'UW'))
    Ng    = 1;
    Hfunc =@(U,lam) (1 - lam)*U(2:end-1) + lam*U(1:end-2);
    name  = 'Upwind';
elseif(strcmp(Scheme,'LF'))
    Ng    = 1;
    Hfunc =@(U,lam) (1 - lam)/2*U(3:end) + (1 + lam)/2*U(1:end-2);
    name  = 'Lax-Friedrichs';
elseif(strcmp(Scheme,'LW'))
    Ng    = 1;
    Hfunc =@(U,lam) (1 - lam^2)*U(2:end-1) + (lam^2-lam)/2*U(3:end)...
                        + (lam^2+lam)/2*U(1:end-2);
    name  = 'Lax-Wendroff';
elseif(strcmp(Scheme,'BW'))
    Ng    = 2;
    Hfunc =@(U,lam) (1 - 3*lam/2 + lam^2/2)*U(3:end-2) ...
```

```
                    + (2*lam - lam^2)*U(2:end-3)...
                    + (-lam + lam^2)/2*U(1:end-4);
    name  = 'Beam-Warming';
else
    error('Unknown Scheme selected!!');
end


% Performing Tasks
% Run scheme
if strcmp(Task,'Solve')
    fprintf('Solving problem with %s scheme\n',name)

    % Discretization
    h  = 0.0025;
    k  = 0.5*h;
    x  = -1:h:1;
    t  = 0:k:Tf;
    N  = numel(x);
    Nt = numel(t);

    % Set initial condition
    U = U0(x,0);

    % Solve using the numerical scheme. We use open boundary conditions
    lam = a*k/h;
    plot_every = 10; % Plot after every 10 time instances

    for i = 2:Nt
        Uext   = [ones(1,Ng)*U(1),U,ones(1,Ng)*U(N)];
        U      = Hfunc(Uext,lam);

        if(mod(i,plot_every)==0 || i==Nt)
            Uexact = U0(x,t(i));
            figure(1)
            plot(x,U,'-r','LineWidth',2)
            hold all
            plot(x,Uexact,'-k','LineWidth',2)
            ylim([-0.5,1.5])
            grid on;
            legend('Numerical','Exact','Location','best')
            title([name,', time = ',num2str(t(i))]);
            hold off
            set(gca,'XTick',-0.5:0.5:1,'FontSize',20)
        end

    end

    % Save plot
    FIG = figure(1);
    fname = sprintf('%s_Profile.pdf',name);
    set(FIG,'Units','Inches');
    pos = get(FIG,'Position');
    set(FIG,'PaperPositionMode','Auto','PaperUnits','Inches','PaperSize',[
       pos(3), pos(4)])
    print(FIG,fname,'-dpdf','-r0')
```

```
% Accuracy test
elseif strcmp(Task,'Acc')
    fprintf('Finding accuracy of %s scheme\n',name)
    H = 0.05*2.^(0:-1:-8);
    E = zeros(numel(H),1);
    n = zeros(numel(H),1);
    % Find Error
    for i = 1:numel(H)

        % Discretization
        h  = H(i);
        fprintf('... Solving for h = %f\n',h)
        k  = 0.5*h;
        x  = -1:h:1;
        t  = 0:k:Tf;
        N  = numel(x);
        Nt = numel(t);

        % Set initial condition
        U = U0(x,0);

        % Exact solution at final time
        Uexact = U0(x,Tf);

        % Solve using the numerical scheme. We use open boundary conditions
        lam = a*k/h;

        for j = 2:Nt
            Uext   = [ones(1,Ng)*U(1),U,ones(1,Ng)*U(N)];
            U      = Hfunc(Uext,lam);
        end

        % Measure error in 1 norm
        E(i) = sum(abs(U-Uexact))*h;
        n(i) = N;
    end
    % Make loglog plot
    p = polyfit(log(n),log(E),1);
    FIG = figure(1);
    loglog(n,E,'-ok');grid on;
    xlabel('Resolution','FontSize',20);
    ylabel('Error','FontSize',20);
    title([name,'. Slope = ',num2str(p(1))],'FontSize',20);
    set(gca,'FontSize',20)
    set(FIG,'Units','Inches');
    pos = get(FIG,'Position');
    set(FIG,'PaperPositionMode','Auto','PaperUnits','Inches','PaperSize',[
       pos(3), pos(4)])
    fname = sprintf('%s_Accuracy.pdf',name);
    print(FIG,fname,'-dpdf','-r0')
end
```