ADDIS ABABA UNIVERSITY
COLLEGE OF NATURAL AND COMPUTATIONAL SCIENCES
DEPARTEMENT OF MATHEMATICS

ON Numerics of one Dimensional Advection Equation

By

Workye Gelaw

A PROJECT SUBMITTED IN PARTIAL FULLFILMENT OF THE REQUIREMENT FOR DEGREE IN MASTERS OF SCIENCE IN MATHEMATICS(DIFFERENTIAL EQUATION)

Addis Ababa, Ethiopia
July, 2020

ADDIS ABABA UNIVERSITY
COLLEGE OF NATURAL AND COMPUTATIONAL SCIENCES
DEPARTEMENT OF MATHEMATICS

The undersigned here by certify that they have read and recommend to the college of Natural and computational Science for acceptance of a project entitled **ON Numerics of one Dimensional Advection Equation** by **Workye Gelaw** in partial fulfillment of the requirement for degree in masters of science in mathematics.

Advisor : Dr. Tadesse Abdi

Examiners

	Name	Signature	Date
1.	Dr.............................	....	....
2.	Dr..............................	....	....

ACKNOWLEDGEMENTS

Foremost, I would like to express my sincere gratitude to God for giving me patience. I am also grateful to my advisor Dr. Tadesse Abdi for the continuous support and giving motivation and immense knowledge in doing my project. Secondly, I would like to extend my thanks to my families and all who encouraged me to complete my project and their heartful help while writing the paper. Lastly, I would like to thank the Department of mathematics, Addis Ababa University, for giving the necessary materials throughout the preparation of my project.

Abstract

This project provides a practical overview of numerical solutions to the linear one dimensional Advection equation using finite difference method. It presents a number of schemes for solution of Linear Advection equation, which are based on the finite difference method, the finite element method and the method of characteristics. It also allow the reader to experiment with the consistency, stability and convergency of finite difference scheme for linear Advection equation and an example with working Matlab code for the scheme is presented.
The aim of this project is to find an optimal and stable solution of a linear advection equation problems that can not be solved analytically easily on a given interval using finite difference methods.

Contents

Notations

$O(h^n)$ - Discretization error or Truncation error.
Δx - The local distance between adjacent points in space.
Δt - The local distance between adjacent time steps.
$u^{(n)}$ - The n^{th} derivative of u

Chapter 1

Introduction to linear Advection equation

In the field of physics, engineering, and earth sciences, Advection is the transport of a substance or quantity by bulk motion. The Advection equation is the partial differential equation that governs the motion of a conserved scalar field as it is Advected by a known velocity vector field. It is derived using the scalar field's conservation law, together with Gauss's theorem, and taking the infinitesimal limit. During Linear advection, a fluid transports some conserved quantity or material via bulk motion. The fluid's motion is described mathematically as a vector field, and the transported material is described by a scalar field showing its distribution over space. Advection requires currents in the fluid, and so cannot happen in rigid solids. It does not include transport of substances by molecular diffusion.

One easily visualized example of Advection is the transport of ink dumped into a river. As the river flows, ink will move downstream in a "pulse" via Advection, as the water's movement itself transports the ink. If added to a lake without significant bulk water flow, the ink would simply disperse outwards from its source in a diffusive manner, which is not Advection. Note that as it moves downstream, the "pulse" of ink will also spread via diffusion. The sum of these processes is called convection. Finite difference techniques were, historically, the most common approach to solving partial differential equations(PDE's) in meteorology.

Hyperbolic systems arise naturally from the conservation laws of physics. Writing down the conservation of mass, momentum and energy yields a system of equations that needs to be solved in order to describe the evolution of the system. In this lecture we will introduce the classical methods for numerically solving such systems.

1.1 Definition and examples of linear Advection equation

Definition 1. *The linear Advection equation, sometimes called the one-way wave equation, is a Hyperbolic partial differential equation. It typically concerns a time* variable t *with one spatial variable* x. Linear Advection equation for u is defined *by:*

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0\,, \begin{cases} -\infty < x < \infty\,, \\ 0 < t, \end{cases} \tag{1.1}$$

where, $u(x,0) = g(x)$ *and* a *is a positive constant.*

The mathematical simplicity of (1.1) should be true the fact that it plays an important role in a wide range of applications. Another application arises in the study of the movement of traffic along a highway. In this case $u(x,t)$ represents the density of cars along the road and a is their speed.

1.2 Method of characterstics of linear Advection equation

The solution of the Linear Advection equation can be written as:

$$\left(\frac{\partial}{\partial t} + a\frac{\partial}{\partial x}\right) u = 0 \tag{1.2}$$

The idea is to transform x, t to new variables r, s in such a way that the derivatives transform as:

$$\frac{\partial}{\partial r} = \frac{\partial}{\partial t} + a\frac{\partial}{\partial x}$$

If this is possible then the above equation becomes $\frac{\partial u}{\partial r} = 0$, and this equation is very easy to solve. With this goal in mind let $x = x(r,s)$, $t = t(r,s)$, in which case using the chain rule, the $r-$derivative transforms as:

$$\frac{\partial}{\partial r} = \frac{\partial x}{\partial r}\frac{\partial}{\partial x} + \frac{\partial t}{\partial r}\frac{\partial}{\partial t}$$

Comparing this with (1.2), we require $\frac{\partial x}{\partial r} = a$ and $\frac{\partial t}{\partial r} = 1$. Integrating these equations yields $x = ar + q(s)$ and $t = r + p(s)$. To determine the s dependence recall that the initial condition specifies the solution along the $x - axis$.
To make it easy to apply the initial condition in the transformed coordinates we ask that the $x - axis$ $(t = 0)$ transform onto the $s - axis$ $(r = 0)$.
Specifically, we require that $r = 0$ imply that $t = 0$ and $x = s$. Setting $r = 0$ and $t = 0$, we conclude $q(s) = s$ and $p(s) = 0$, and so the change of variables
We are looking for is: $x = ar + s$, $t = r$.
Inverting this transformation, one obtains $r = t$ and $s = x - at$. We are now able to write (1.1) as $\frac{\partial u}{\partial r} = 0$, which means that $u = u(s) = u(x - at)$. With the initial condition we therefore conclude that the solution of the advection equation is:

$$u(x,t) = g(x - at). \tag{1.3}$$

This is a traveling wave that moves with speed a. It is also seen that the solution is constant along lines of the form $x - at = constant$. These lines are called characteristics for the equation and the method we used to find the solution is known as the method of characteristics.

Example 1.

$g(x)= \begin{cases} 1, 0 \leq x \leq 1, \\ 0, otherwise \end{cases}$

From (1.3) the solution is:

$u(x,t) = \begin{cases} 1, 0 \leq x - at \leq 1, \\ 0, otherwise \end{cases}$

or equivalently,

$u(x,t) = \begin{cases} 1, at \leq x \leq 1 + at, \\ 0, otherwise \end{cases}$

Example 2. *Solve the following linear advection equation using analytically*

- $u_t + 2u_x = 0$. From equation(1.3) we have $a = 2$. And the general analytic solution for the given equation becomes: $u(x,t) = g(x - 2t)$.
- $2u_t - u_x = 0$. By rearranging of the equation we get: $u_t - \frac{1}{2}u_x = 0$. $a = -\frac{1}{2}$.
Then the general analytic solution for the given equation becomes:
$u(x,t) = g(x + \frac{1}{2}t)$.
- $u_t + 5u_x = 0$, $u(x,0) = e^x$. From equation(1.3) we get $a = 5$. and the general solution of the given equation becomes $u(x,t) = g(x - 5t)$.
Using the initial condition $u(x,0) = e^x$, we get the general solution $u(x,t) = e^{x-5t}$.

1.3 Boundary Conditions

The unbounded spatial interval used in (1.1) is not going to make it into our numerical algorithms which we will discuss below. To compute the solution we need to specify left (x_L) and right (x_R) endpoints for the spatial grid. In this case the grid points are $x_i = x_L + ih$, where $i = 0, 1, 2, ..., N + 1$ and $x_N + 1 = x_R$.

1.4 Finite Differnce Method

In mathematics, finite-difference methods (FDM) are numerical methods for solving differential equations by approximating them with difference equations, in which finite differences approximate the derivatives. FDMs are thus discretization methods. The finite difference approach is one of the premier mathematical tools employed to solve partial differential equations. It is a means of obtaining numerical solutions to partial differential equations.
The most common finite difference methods for the solution of partial differential equation which we use are:

- Explicit method
- Implicit method
- Lax-wendroff method

These are closely related but differ in stability, accuracy and execution speed. In the formulation of a partial differential equation problem, there are three components to be considered:

- The partial differential equation.
- The region of space-time on which the partial differential equation is required to be satisfied
- The boundary and initial conditions to be met

The application of finite difference method to a particular differential equation problem includes the following steps:

- Construction of a discrete finite difference model of the problem:
  - coverage of the computational domain by a space-time grid,
  - construction of a system of finite difference equations using different methods
- Analysis of the finite difference model
- Numerical computations using Matlab

Chapter 2

First-Order Methods

2.1 Grid(Mesh) point

The mesh (grid) is the set of locations where the discrete solution is computed. These points are called nodes, and if one were to draw lines between adjacent nodes in the domain the resulting image would resemble a net or mesh. Two key parameters of the mesh are Δx the local distance between adjacent points in space, and Δt, the local distance between adjacent time steps Obtain a system of equations with finite dimension, we must solve the equation on some bounded domain. The domain is partitioned in space and in time and approximations of the solution are computed at the space or time points.

One way to numerically solve Linear Advection equation is to approximate all the derivatives by finite differences. We partition the domain in space using a mesh $x_0, ..., x_L$ and in time using a mesh $t_0, ..., t_M$.
We will compute the solution over the two-dimensional region $x_L < x < x_R$, $0 < t \leq T$, where x_L , x_R , T are specified at the time the computation is carried out. The discretization of the given differential equation is obtained by dividing the given domain into a finite number of elements. We assume a uniform partition both in space and in time, so the difference between two consecutive space points will be h and between two consecutive time points will be k. The domain is partitioned in space and in time and approximations of the solution are computed at the space or time points.

The usual formulas are used, namely,

$$t_j = jk,$$

$$x_i = x_L + ih$$

for $j = 0, 1, 2, ..., M$ and for $i = 0, 1, 2, ..., N + 1$,
where $k = T/M$ and $h = (x_R - x_L)/(N + 1)$.
Here, Δx is usually called grid spacing, and Δt is called time step since t usually represents time.
At the grid points a function $u(x, t)$ is to be approximated by a grid function $u(x_i, t_j)$. A value of $u(x_i, t_j)$ can be denoted by u_{ij}.

2.2 First order forward and backward finite difference methods

Accuracy of a finite difference formula is a fundamental issue when discretizing differential equations. Truncation (or discretization) error is caused when approximations are used to estimate some quantity. The error between the numerical solution and the exact solution is determined by the error that is committed by going from a differential operator to a difference operator.
As $(\Delta x, \Delta t) \longrightarrow (0, 0)$ the numerical solution obtained with any useful scheme will approach the true solution to the original differential equation. However, the rate at which the numerical solution approaches the true solution varies with the scheme.
The principle of finite difference methods is close to the numerical schemes used to solve ordinary and partial differential equations.
Finite difference methods consists in approximating the differential operator by replacing the derivatives in the equation using differential quotients.
The domain is partitioned in space and in time and approximations of the solution are computed at the space or time points.
Now, using the derivative of a function $u(x, t)$ at h gives:

$$u'(x) = \lim_{h \to 0} \frac{u(x + h, t) - u(x, t)}{h}$$

If h is sufficiently small then the derivative of $u(x)$ becomes:

$$u'(x) \approx \frac{u(x + h, t) - u(x, t)}{h} \tag{2.1}$$

Now consider the **Taylor series**, a way to approximate the value of a function by taking the sum of its derivatives at a given point, expansion of a function around a point $x = x_0$.
The Taylor series expansion of a function $u(x)$ at $x = x_0$ is given by the formula:

$$u(x) = \sum_{n=0}^{\infty} \frac{u^{(n)}(x_0)}{n!}(x - x_0)^n \tag{2.2}$$

where,
- $u^{(n)}(x_0) = \frac{d^n u}{dx^n}$ at $x = x_0$
- $u^0(x_0) = u(x_0)$

If we let $x = x_0 + h$, then $x - x_0 = h$, and the series can be written as follows

$$\begin{aligned} u(x_0 + h) &= \sum_{n=0}^{\infty} \frac{u^{(n)}(x)}{n!} h^n \\ &= u(x_0) + \frac{u'(x_0)}{1!} h + \mathcal{O}(h^2) \end{aligned}$$

where, $\mathcal{O}(h^2)$ is the remaining terms of the series with leading term of order the error h^2 incurred in neglecting this part of the series expansion when calculating $u(x_0 + h)$.

Now, we will replace the first order $\frac{du}{dx}$ at $x = x_0$, by the expression $u'(x) \approx \frac{u(x_0+h)-u(x_0)}{h}$, and so on and now selecting an appropriate value for h, and indicating that the error introduced in the calculation is of order h.

Now, from the above equation and using **Taylor** series expansion for first order derivatives for $h > 0$:

$$u(x_0 + h) = u(x_0) + u'(x_0)h + \mathcal{O}(h^2)$$

And for $h < 0$ the Taylor series expansion becomes:

$$u(x_0 - h) = u(x_0) - u'(x_0)h + \mathcal{O}(h^2)$$

Rearranging the above equations for solving first order and second order derivative with Taylor series gives the following results:

$u'(x_0) = \frac{u(x_0+h)-u(x_0)}{h} + \mathcal{O}(h)$, for $h > 0$
$u'(x_0) = \frac{u(x_0)-u(x_0-h)}{h} + \mathcal{O}(h)$, for $h < 0$

For small space size h then $\mathcal{O}(h) \to 0$ and the above equations become:

$$u'(x_0) \approx \frac{u(x_0 + h) - u(x_0)}{h} \tag{2.3}$$

Equation (2.3) is called a first order forward finite difference approximation to $u'(x_0)$. This approximation is named as forward finite difference approximation since we start at x_0 and step forwards to the point $x_0 + h$.

$$u'(x_0) \approx \frac{u(x_0) - u(x_0 - h)}{h} \tag{2.4}$$

Equation (2.4) is called the backward difference formula because it involves the values of u at x_0 and $x_0 - h$.
The order of magnitude of the truncation error for the backward difference approximation is the same as that of the forward difference approximation.

Next, To obtain first order central difference method adding equation (2.3) and equation (2.4) and we get:
$u(x_0 + h)$ -$u(x_0 - h)$ $=2hu'(x_0) + \mathcal{O}(h^2)$

$$u'(x_0) \approx \frac{u(x_0 + h) - u(x_0 - h)}{2h} \tag{2.5}$$

This is the central difference approximation to $u'(x_0)$.

Note that in order to find the first order forward difference, first order backward difference and first order central difference method with respect to time(t) we use the same method as we have seen above with respect to space(h).

Chapter 3

Finite difference methods in solving Linear Advection equation

The derivation of finite difference schemes(FDS) for solving the linear advection equation involves approximating the partial derivatives.

3.1 Upwind Scheme

There are many different discretizations for the linear advection equation: Here we will look at some of the most important principles. Because the advection equation describes a transport phenomenon, from $x = 0$ to $x = 1$ if $a > 0$, it is important that the scheme picks up this behavior correctly.

Equation(1.1) describes a wave propagating along the $x - axis$ with a velocity a. This equation is also a mathematical model for one-dimensional linear Advection. Consider a typical grid point i in the domain. In a one-dimensional domain, there are only two directions associated with point $i - left$ (towards negative infinity) and right (towards positive infinity). If a is positive, the travelling wave solution of the equation above propagates towards the right, the left side of i is called upwind side.

There are many different discretizations for the linear advection equation. Now, from those discretizations for upwind scheme, we use a forward difference in time and a backward difference in space of using the above equations and substituting in (1.1) we have the following result:

$$u_t + au_x = 0$$

$$\frac{u(x_i, t_{j+1}) - u(x_i, t_j)}{k} + a\frac{u(x_i, t_j) - u(x_{i-1}, t_j)}{h} + \mathcal{O}(h) + \mathcal{O}(k) = 0$$

$$\frac{u(x_i,t_{j+1})-u(x_i,t_j)}{k}+a\frac{u(x_i,t_j)-u(x_{i-1},t_j)}{h}+\tau_{ij}=0$$

Where $\tau_{ij}=\mathcal{O}(h)+\mathcal{O}(k)$.
Dropping the truncation error and rearranging the terms we obtain what is known as the upwind scheme:

$$u_{i,j+1}=(1-\lambda)u_{ij}+\lambda u_{i-1,j}, for \begin{cases} i=1,2,3,...,N+1, \\ j=0,1,2,...,M-1, \end{cases} \tag{3.1}$$

Where, $\lambda=\frac{ak}{h}$.

Example 3. *Solve $u_t+4u_x=0$, with the initial conditions.*
$u(x,0)=x(4-x)$ and $u_t(x,0)=0$, We have $a=4$.

let us assume that we use an upwind scheme method with $h=2$ and $k=0.5$.
Let the number of time steps up to which the computations are to be performed be 4. Then,we have $\lambda=\frac{ak}{h}=\frac{2(0.5)}{1}=1$.
Now using an upwind scheme formula is given by equation(3.1) we get:
$u_{i,j+1}=(1-\lambda)u_{ij}+\lambda u_{i-1,j}=\lambda u_{i-1,j}$, $i=1,2,3$ and $j=1,2,3$
The boundary conditions give the values $u(0,j)=0=u(4,j)$, for all j and the initial conditions give the following values.

$$u(x,0)=x(4-x), u_{0,0}=0, u_{1,0}=3$$
$$u_{2,0}=u(2,0)=4, u_{3,0}=3, u_{4,0}=0$$

Now using first order central difference approximation method to $u_t(x,0)=0$ which gives $u_{i,-1}=u_{i,1}\Rightarrow u_{i,1}=u_{i-1,0}$. We have the following results:
For $j=1$: to give $u_{i,j+1}=\lambda u_{i-1,j}$.

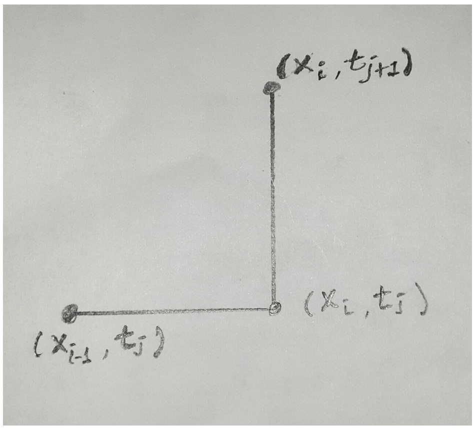
- $i=1: u_{1,2}=u_{0,1}=0$,
- $i=2: u_{2,2}=u_{1,1}=3$,
- $i=3: u_{3,2}=u_{2,1}=4$.

For $j=2$ to calculate $u_{i,3}=u_{i-1,2}$.

- $i=1: u_{1,3}=u_{0,2}=0$,
- $i=2: u_{2,3}=u_{1,2}=3$,
- $i=3: u_{3,3}=u_{2,2}=4$.

For $j = 3$ to calculate $u_{i,4} = u_{i-1,3}$.

- $i = 1 : u_{1,4} = u_{0,3} = 0$,
- $i = 2 : u_{2,4} = u_{1,3} = 3$,
- $i = 3 : u_{3,4} = u_{2,3} = 4$.

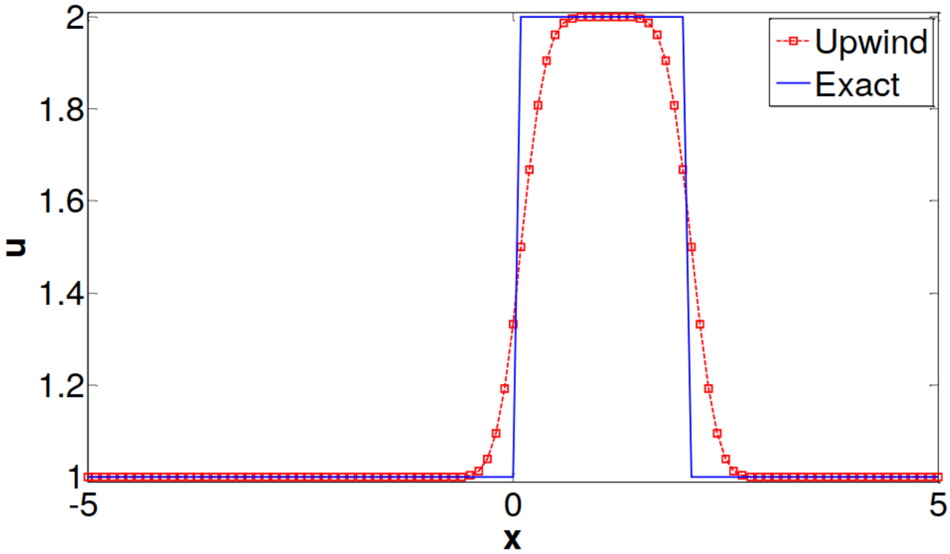


Stencils for the upwind scheme.

Matlab Code for Upwind Scheme

```
clc;
clear all;
close all;
a = 1;
j = 0;
for j=100:100:1000
    j
%j = j+1;
h = 10./j;
k = 0.5*h;
n = j+1;
x = h*[0:j]-5;
for i=1:n
    uexact(i) = 1 + H(x(i)+1-1) -H(x(i)-1-1);
end
for i=1:n
    u(i) = 1 + H(x(i)+1) -H(x(i)-1);
    end
    uo = u;
for t = 0:k:1
     for i = 2:n-1
         u(i) = uo(i) -a*k*(uo(i)-uo(i-1))/h;
     end
     uo = u;
end
plot(x,u,'--rs','LineWidth',2)
hold on;
plot(x,uexact,'-b','LineWidth',2)
axis([-5 5 0.99 2.01])
set(gca,'FontSize',30);
xLabel('x','FontSize',30,'fontweight','b');
yLabel('u','FontSize',30,'fontweight','b');
legend('Upwind','Exact');hold offpause(0.1);
error(j/100) = sum(abs(u-uexact))/j;
end
figure;
loglog(10./([100:100:1000]),error,'--r','LineWidth',3);
hold on
```

```
set(gca,'FontSize',30);
xLabel('h','FontSize',30,'fontweight','b');
yLabel('error','FontSize',30,'fontweight','b');
```



3.2 Downwind Scheme

Now from those discretizations for downwind scheme we use the first order forward differences for both space and time and substituting in (1.1) we have the following result:

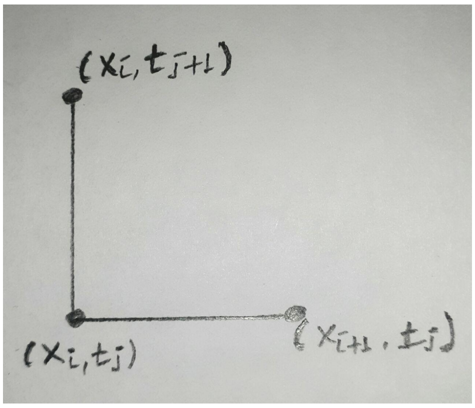
$$u_t + au_x = 0$$

$$\frac{u(x_i,t_{j+1}) - u(x_i,t_j)}{k} + a\frac{u(x_{i+1},t_j) - u(x_i,t_j)}{h} + \mathcal{O}(h) + \mathcal{O}(k) = 0$$

$$\frac{u(x_i,t_{j+1}) - u(x_i,t_j)}{k} + a\frac{u(x_{i+1},t_j) - u(x_i,t_j)}{h} + \tau_{ij} = 0$$

Where, $\tau_{ij} = \mathcal{O}(h) + \mathcal{O}(k)$.
Now dropping the truncation error and rearranging the terms, one obtains the downwind scheme, given as:

$$u_{i,j+1} = (1+\lambda)u_{ij} - \lambda u_{i+1,j}, for \begin{cases} i = 0, 1, 2, 3, ..., N \\ j = 0, 1, 2, ..., M-1, \end{cases} \tag{3.2}$$



Stencils for the downwind scheme.

Downwind Scheme Matlab Code

```
% [u,err,x,t] = explicit_downwind(t_0,t_f,M,N)
%
% solves the heat equation u_t + u_x = 0 [0,2*pi]
%
% If I'm going to use the FFT to analyse the solution then I'd like to take
```

```
% N of the form 2^k.  This will also have the advantage of making sure there's
% always a meshpoint at x = pi

function [u,x,t] = explicit_downwind(t_0,t_f,M,N)

% define the mesh in space
dx = 2*pi/N;
x = 0:dx:2*pi;
x = x';

% define the mesh in time
dt = (t_f-t_0)/M;
t = t_0:dt:t_f;

r = dt/dx;

% choose the wave number of the initial data and give its decay rate
u = zeros(N+1,M+1);
u(:,1) = x<=pi;

% I want to do the unstable forward-time, forward space scheme:
%
% u_new(j) = u_old(j) - r*(u_old(j+1)-u_old(j))

% for j=1:M
%     u(:,j)'
%     u(1:N,j+1) = u(1:N,j)-r*diff(u(:,j));
%     u(N+1,j+1) = 0;
% end

for j=1:M
    for k=1:N
        u(k,j+1) = u(k,j)-r*(u(k+1,j)-u(k,j));
    end
    u(N+1,j+1)=0;
end
```

3.3 Lax–Wendroff Method

The Lax–Wendroff method, named after **Peter Lax** and **Burton Wendroff**, is a numerical method for the solution of partial differential equations, based on finite difference methods. The upwind scheme in the previous section is only first order accurate.
For practical problems, however, first order accuracy is often not enough. There are various schemes that give higher-order results, the best known being the Lax-Wendroff scheme.
One of the most well known second order schemes for approximating the advection equation is the Lax-Wendroff scheme and is derived as follows:

$$u_{i,j+1} = Au_{i+1,j} + Bu_{ij} + Cu_{i-1,j}. \tag{3.3}$$

Note: equation(3.3) is called upwind scheme if A=0 and Downwind scheme if C=0.

The coefficients A, B and C are constants and determined by requiring that the truncation error associated with this approximation.
To calculate the error note that $u_t = -au_x$, $u_{tt} = -au_{xt} = -a(u_t)_x = a^2u_{xx}$, and $u_{ttt} = -a^3u_{xxx}$.
With this and Taylor's theorem, one finds that:

$$\begin{aligned} u(x_i, t_{j+k}) &= u(x_i, t_j) + ku_t(x_i, t_j) + \frac{1}{2}k^2u_{tt}(x_i, t_j) + \frac{1}{6}k^3u_{ttt}(x_i, t_j) + ... \\ &= u(x_i, t_j) - aku_x(x_i, t_j) + \frac{1}{2}a^2k^2u_{xx}(x_i, t_j) - \frac{1}{6}a^3k^3u_{xxx}(x_i, t_j) + ... \end{aligned}$$

Similarly,

$$u(x_{i\pm h}, t_j) = u(x_i, t_j) \pm hu_x(x_i, t_j) + \frac{1}{2}h^2u_{xx}(x_i, t_j) \pm \frac{1}{6}h^3u_{xxx}(x_i, t_j) + ...$$

Now, when $u(x_i, t_j)$ and the truncation error τ_{ij} are in the difference equation, the form of the equation is:

$$u(x_i, t_{j+k}) = Au(x_{i+h}, t_j) + Bu(x_i, t_j) + Cu(x_{i-h}, t_j) + k\tau_{ij}$$

$$\begin{aligned} \tau_{ij} &= \frac{1}{k}(u(x_i, t_{j+k}) - Au(x_{i+h}, t_j) - Bu(x_i, t_j) - Cu(x_{i-h}, t_j)) \\ &= \frac{1}{k}(1 - A - B - C)u(x_i, t_j) - \frac{h}{k}(A - C + \lambda)u_x(x_i, t_j) - \frac{1}{2}\frac{h^2}{k}(A + C - \lambda^2)u_xx(x_i, t_j) \\ &\quad -\frac{1}{6}\frac{h^3}{k}(A - C + \lambda^3)u_{xxx}(x_i, t_j) + ... \end{aligned}$$

Note that as τ_{ij} must go to zero as h and k go to zero and the coefficients goes to zero in the above as possible and we get the result:

$$\begin{aligned}u &: 1 - A - B - C = 0,\\ u_x &: \lambda + A - C = 0,\\ u_{xx} &: -\lambda^2 + A + C = 0.\end{aligned}$$

Solving these equations and we get the result:

$$u_{i,j+1} = -\frac{1}{2}\lambda(1-\lambda)u_{i+1,j} + (1-\lambda^2)u_{ij} + \frac{1}{2}\lambda(1+\lambda)u_{i-1,j}. \tag{3.4}$$

The truncation error in this case is:

$$\begin{aligned}\tau_{ij} &= -\frac{1}{6}\frac{h^3}{k}(A - C + \lambda^3)u_{xxx}(x_i, t_j) + \ldots\\ &= \frac{1}{6}(-k^2a^2 + ah^2)u_{xxx}(x_i, t_j) + \ldots\\ &= \omega(k^2) + \omega(h^2).\end{aligned}$$

Lax–Wendroff method Matlab Code

```
% [u,err,x,t] = lax_wendroff(@f,t_0,t_f,M,N)
%
% solves the heat equation u_t + u_x = 0 [0,2*pi]
%
% If I'm going to use the FFT to analyse the solution then
    I'd like to take
% N of the form 2^k.  This will also have the advantage of
    making sure there's
% always a meshpoint at x = pi
%
% f.m is the function that contains the initial data

function [u,u_exact,x,t] = lax_wendroff(f,t_0,t_f,M,N)

% define the mesh in space
dx = 2*pi/N;
x = 0:dx:2*pi;
x = x';

```

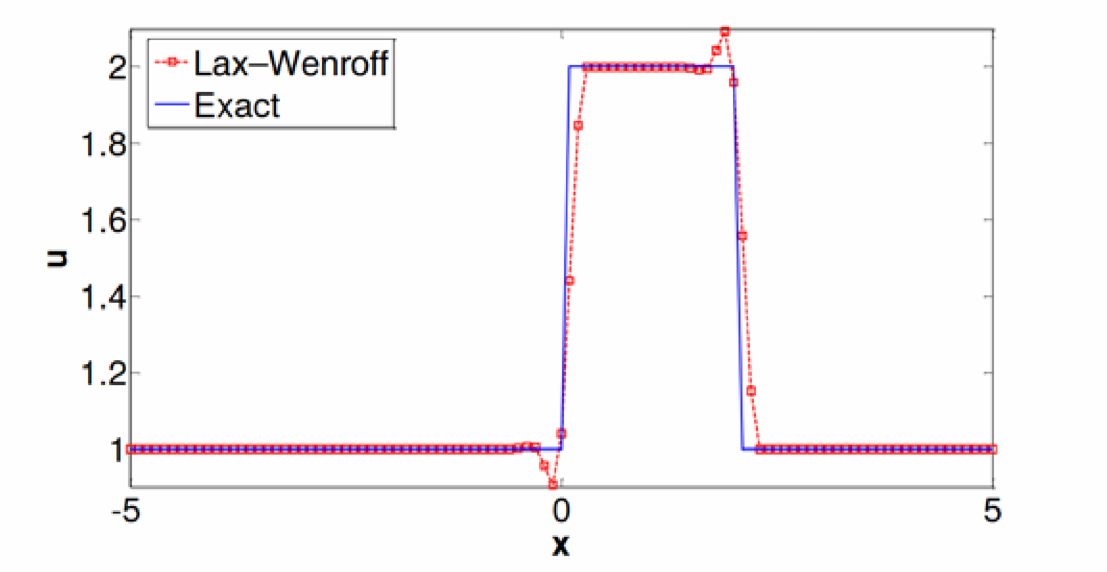
```
% define the mesh in time
dt = (t_f-t_0)/M;
t = t_0:dt:t_f;

c = 1/2;
% display('this number should be between -1 and 1 for stability:')
mu = c*dt/dx;

% choose the wave number of the initial data and give its decay rate
u = zeros(N+1,M+1);
u(:,1) = f(x);
u_exact(:,1) = u(:,1);

% I want to do the Lax Wendroff scheme:
%
% u_new(j) = u_old(j) - (mu/2)*(u_old(j+1)-u_old(j-1))
%                                + (mu^2/2)*(u_old(j-1)-2*u_old(j)+u_old(j+1))

for j=1:M
    for k=2:N
        u(k,j+1) = u(k,j) - (mu/2)*(u(k+1,j)-u(k-1,j)) + (mu^2/2)*(u(k+1,j)-2*u(k,j)+u(k-1,j));
    end
    % I code in the exact values at the endpoints.
    u(1,j+1)=1;
    u(N+1,j+1)=0;
    X = x-c*t(j+1);
    u_exact(:,j+1) = f(X);
end
```



3.4 Implicit Method

Implicit methods playe a central role in solving partial differential equations(PDEs). In order to illustrate implicit method we use a backward difference in time and a centered difference in space). The resulting finite difference equation becomes:

$$u_t(x_i, t_j) + au_x(x_i, t_j) = 0$$

$$\frac{u(x_i, t_j) - u(x_i, t_{j-1})}{k} + a\frac{u(x_{i+1}, t_j) - u(x_{i-1}, t_j)}{2h} = 0$$

Rearranging and collecting like terms, we get:

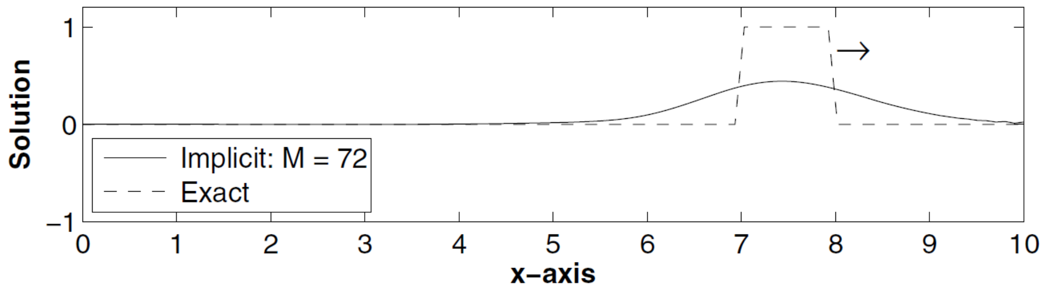
$$\lambda u_{i+1,j} + 2u_{i,j} - \lambda u_{i-1,j} = 2u_{i,j-1}. \tag{3.5}$$

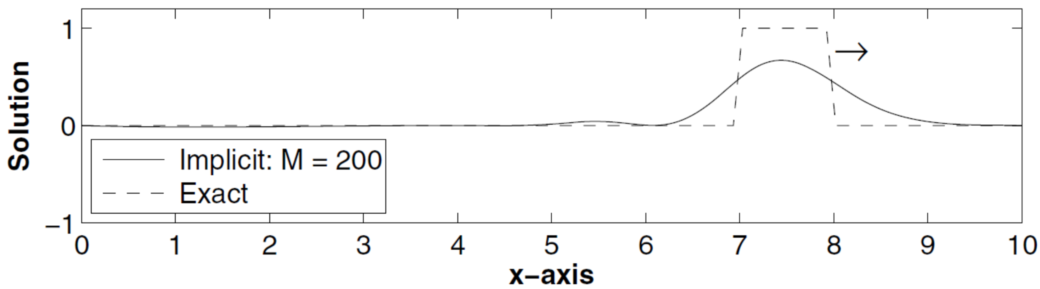
The implicit nature of the approximation means that the numerical solutions at time level $t = t_j$ depend on all values at $t = t_{j-1}$.
Consequently, the numerical domain of dependence of (x_i, t_j) is the entire x-axis, and this clearly satisfies the Courant-Friedrichs-Lewy (CFL) condition which we will discuss below.

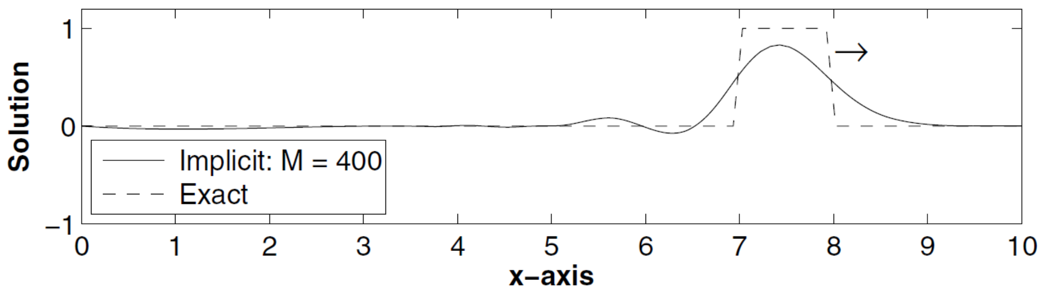
Example 4. *Solve the linear advection eqation given below with $a = 1$ in the range for $0 \leq x \leq 10$, $0 \leq t \leq 7$. Use 100 points along the x-axis. For the number of* time points we try M = 72 *($\lambda = 0.98$), $M = 100$ ($\lambda = 0.7$), $M = 200$, and $M = 400$.*

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

The numerical result of the given example with the conditions $M = 72$, $M = 200$ and $M = 400$ are the following:







In general, the above methods we discussed can be summarized simply as follows

methods for solving	$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0$		
Using	$u_{i,j+1} = Au_{i+1,j} + Bu_{i,j} + Cu_{i-1,j}$		
Method	Coefficients	Truncation Errors	CFL Conditions
upwind	$A = 0, B = 1 - \lambda, C = \lambda$	$O(h) + O(k)$	$\lambda \leq 1$
Lax–Wendroff	$A = -\frac{\lambda}{2}(1-\lambda), B = 1 - \lambda^2, C = \frac{\lambda}{2}(1+\lambda)$	$O(h^2) + O(k^2)$	$\lambda \leq 1$

3.5 Numerical Domain of Dependence for Advection equation

3.5.1 The Courant-Friedrichs-Levy (CFL) Condition

The Courant-Friedrichs-Levy (CFL) Condition comes from work done around 1928which used finite difference methods to prove the existence of solutions of certain PDEs. The CFL condition states that a necessary condition for the convergence of a numerical method is that the numerical domain of dependence contains the analytical domain of dependence. It is important to note that the CFL condition is not a sufficient condition for the convergence of the approximate solution to theexact solution.

The domain of dependence of the solution of a PDE at position x_i and time t_j is the set of points at a previous time that influence the solution at position x_i and time t_j.
Given a spatial location $x = x_i$, and time $t = t_j$, the grid points along the x-axis that contribute to the solution at (x_i, t_j) form what is known as the numerical domain of dependence for the solution at (x_i, t_j).

For the linear Advection equation the domain of dependence for (x_i, t_j) is the single point $(x_0, 0)$, where $x_0 = x_i - at_j$.
In other words, the exact solution at (x_i, t_j) is determined solely by the value of the initial condition at $x_0 = x_i - at_j$.
If the computed solution is to have any hope of producing the correct answer, it is essential that the numerical domain of dependence bound the domain of dependence. This gives rise to the following principle.

3.6 Von Neumann (or Fourier) Analysis of Stability

3.6.1 Stability

A numerical solution is said to be stable if it does not magnify the errors that appear in the course of numerical solution process. Stability guarantees that the method produces a bounded solution whenever the solution of the exact equation is bounded.

The numerical domain of dependence must bound, or contain, the domain of dependence for the problem.
The stability of a finite difference schemes for hyperbolic partial differential equations can be analyzed by the Von Neumann or Fourier method.
The stability analysis applies directly to the methods used to solve the Advection equation. Assume that the solution equation (1.1) be of the form :

$$u_{ij} = w_j e^{rx_i I} \tag{3.6}$$

where, $I = \sqrt{-1}$. The function w_j is determined from the difference equation, and the requirement for stability is that w_j remain bounded as j increases. Now for upwind scheme in equation(3.1) substitute in equation(3.6) on upwind scheme method and we get the result:

$$\begin{aligned} u_{i,j+1} &= (1-\lambda)u_{ij} + \lambda u_{i-1,j} \\ w_{j+1}e^{rx_i I} &= (1-\lambda)w_j e^{rx_i I} + \lambda w_j e^{rx_{i-1} I} \end{aligned}$$

Cancelling terms and using the half angle formulae we get:

$w_j = k^j w_0$,
where $k = 1 - \lambda + \lambda\cos(rh) - I\lambda\sin(rh)$.

$$\begin{aligned} k^2 &= (1-\lambda+\lambda cos(rh) - I\lambda sin(rh))(1-\lambda+\lambda cos(rh) + I\lambda sin(rh)) \\ &= 1 - 2\lambda + 2\lambda cos(rh) - 2\lambda^2 cos(rh) + 2\lambda^2 \\ &= 1 - 2\lambda(1 - cos(rh)) - 2\lambda^2(cos(rh) - 1) \end{aligned}$$

Now using half-angle formula $cos(rh) = cos(\frac{rh}{2} + \frac{rh}{2}) = cos^2(\frac{rh}{2}) - sin^2(\frac{rh}{2})$ and

$\cos^2(\frac{rh}{2}) = 1 - \sin^2(\frac{rh}{2})$, we get:

$$\begin{aligned} k^2 &= 1 - 2\lambda(2\sin^2(\frac{rh}{2})) - 2\lambda^2(-2\sin^2(\frac{rh}{2})) \\ &= 1 - 4\lambda\sin^2(\frac{rh}{2}) + 4\lambda^2\sin^2(\frac{rh}{2}) \\ &= 1 - 4\lambda(1-\lambda)\sin^2(\frac{rh}{2}) \end{aligned}$$

Now for the salution to be stable and by solving the inequality, divide both sides by $\lambda\sin^2(\frac{rh}{2})$ we get:

$0 \leq 1 - \lambda$ and which gives $0 \leq \lambda \leq 1$, scince $\lambda \geq 0$

To guarantee that w_j remains bounded we require $|k|^2 \leq 1$. This implies that the condition $0 \leq \lambda(1-\lambda)\sin^2(\frac{rh}{2})$. This holds, irrespective of the value of r, if $\lambda \leq 1$, and this is the stability condition for the upwind scheme method.

In a similar way for lax-wendroff method for stability condition substitute equation (3.4) in equation (1.1) and we get the stability condition $\lambda \leq 1$.

When choosing a method for solving a differential equation problem it is necessary to have some knowledge about how to analyze the result.
The solution method should have certain properties. In most cases, it is not possible to analyze the complete solution method. The most important properties are summarized below.

Consistency is used to indicate the accuracy of the method. Consider the expression of equation(2.3) to follow the consistency condition that is satisfied by:

$$\tau_{ij} \to 0 \text{ as } h \to 0 \text{ and } k \to 0.$$

Since finite difference of a partial differential equation is consistent, if the difference between partial differential equation and finite differential equation vanishes as the space and time step size approach zero.

Consistency deals with how well the finite difference equation approximates the partial differential equation and it is the necessary condition for convergence.

Summary

Using an finite difference numerical scheme, Upwind, Downwind, Lax-wendroff, for the first order time and space derivatives we derived a difference equation equivalent to the linnear Advection equation up to an error of order (Δx) , which lead to analyzing the stability of this scheme we arrived at the condition $a\Delta t \leq \Delta x$.

We also observed that the those derived schemes uses two previous time steps to compute the values of the numerical solution at a particular grid point, thus one needs the values of initial values to run the scheme. These we were able to find from the initial conditions by finite difference, Upwind, Downwind, Lax-wendroff e approximation,approximation methods which does not add smaller order errors to the numerical scheme.

Bibliography

[1] J.W. Thomas, *Numerical partial differential equations: Finite difference method*, Springer, Verlag, Inc, New York,1995.

[2] Mark H. Holmes, *Introduction to Numerical methods in Differential equations*, Troy, New York, 2006.

[3] Atkison, K, Elementary Numerical Analysis, Wiley, New York, 1995

[4] Peter J. Olver, *Numerical Analysis Lecture Notes on differential equations*, 2008