

# Integration of Partial Differential Equations

Erik Jansen, Hans van Piggelen en Marcel Haas

## Summary

Much equations encountered in physics are *partial differential equations*, particularly in fluid mechanics. Such equations can in general not be solved analytically, so numerical methods must be used. As a simple example the *Linear Advection Equation* is solved using the *Lax-Friedrichs scheme*, also presented in the text.

## 1 Introduction

In most equations of physics one must deal with partial differential equations (the equations in fluid dynamics are an obvious example).

### 1.1 The Concept of a Spatial Grid

When we integrate the ordinary differential equation  $du/dt = F$  numerically, we only take snapshots at discrete times. If we want to solve partial differential equations the concept of taking snapshots works exactly the same for the time-dependence. However, we also want to know the spatial dependence.

Just like we have to take snapshots in time, we can only determine  $u(x, t)$  at discrete points in space. These discrete points define a **grid** and a numerical scheme only produces values of  $u(x, t)$  for points on the grid and for discrete times. We can keep notation simple by using the following conventions:

$$u(x = x_j, t = t_n) = u_j^n \quad (1)$$

If one wants to find a value of  $u(x, t)$  at some  $x$  or  $t$  that does not coincide with the grid one has to interpolate between values at the neighboring grid points.

### 1.2 Derivatives

Since a numerical scheme only gives you values for discrete points in space and time, we have to find a numerical approximation for the spatial derivatives  $\partial u/\partial x$ ,  $\partial^2 u/\partial x^2$  etc. Here we will limit the discussion to the first two derivatives with respect to  $x$ , assuming for simplicity that the grid is a *uniform grid*:

$$x_{j+1} - x_j = \Delta x = \text{constant for all } j.$$

Often one chooses to give a numerical estimate for the derivatives which uses values of  $x_j^n$  centered around the grid point in question:

$$\left(\frac{\partial u}{\partial x}\right)_{x=x_j, t=t_n} \approx \frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x} \quad (2)$$

Here the values of the  $u^n$  on the two grid points on either side are used to calculate the derivative.

In a similar fashion, a good approximation for the second derivative is:

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_{x=x_j, t=t_n} \approx \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta x^2} \quad (3)$$

Equations 2 and 3 are both second-order accurate in space, meaning that the difference between the numerical estimates and the true values are of order  $\Delta x^2$ .

### 1.3 Stability and the Von Neumann Criterion

Numerical schemes used to solve partial differential equations should converge to the true solution for small spatial and time steps, and they should be stable so that small errors are not amplified to the point that the true solutions is completely washed out.

There is a simple method to check the *linear* stability of a scheme, which was invented by Von Neumann. It makes use of the Fourier Theorem:

$$u(x_j, t) = \sum_k \hat{u}(t) e^{ikx_j} \quad (4)$$

Note that the spatial dependence of each individual Fourier mode is prescribed:

$$e^{ikx_j} = \cos(kx_j) + i \sin(kx_j) \quad (5)$$

The strength of Von Neumann's stability criterion is that it can be applied to each Fourier mode separately. It simply states that a scheme is linearly stable provided the condition

$$|g_k(\Delta x, \Delta t)| = \sqrt{g_k(\Delta x, \Delta t) g_k^*(\Delta x, \Delta t)} \leq 1 \quad (6)$$

where  $g_k$  (the amplification factor) is defined in:

$$g_k = \frac{\hat{u}_k^{n+1}}{\hat{u}_k^n}. \quad (7)$$

## 2 The Linear Advection Equation

In this section the linear advection equation is considered:

$$\frac{\partial u}{\partial t} = -a \frac{\partial u}{\partial x} \quad (8)$$

where the advection velocity  $a$  is considered constant. Equation 8 can be rewritten in dimensionless form as

$$\frac{\partial u}{\partial \tilde{t}} + \frac{\partial u}{\partial \tilde{x}} = 0 \quad (9)$$

if the dimensionless time and space variables are defined as  $\tilde{t} = at/L$ ,  $\tilde{x} = x/L$ , with  $L$  some arbitrary normalizing length, as can easily be seen by in, after which equation 8 again follows for the advection.

If a function of the form  $u(x, t) = F(x - at) \equiv F(\xi)$  with  $\xi = x - at$  is defined, it can be shown to be a solution to the advection equation (8):

$$\frac{\partial F(\xi)}{\partial t} + a \frac{\partial F(\xi)}{\partial x} = 0 \quad (10)$$

$$-aF(x - at)F'(x - at) + aF(x - at)F'(x - at) = 0 \quad (11)$$

### 2.1 The Lax-Friedrichs Method

The *Lax-Friedrichs method* tries to solve the linear advection equation using the scheme

$$u_j^{n+1} = \frac{1}{2}(u_{j-1}^n + u_{j+1}^n) - \frac{a\Delta t}{2\Delta x}(u_{j+1}^n - u_{j-1}^n) \quad (12)$$

A Fourier mode in this scheme has an amplification factor (which follows from Equation 7):

$$g_k(\Delta x, \Delta t) = \cos(k\Delta x) - i \frac{a\Delta t}{\Delta x} \sin(k\Delta x) \quad (13)$$

Von Neumann's stability criterion (Equation 6) leads to the *Courant-Friedrichs-Lewy condition* for the allowed time step of this scheme:

$$\left| \frac{a\Delta t}{\Delta x} \right| \leq 1 \iff \Delta t \leq \frac{\Delta x}{|a|} \quad (14)$$

### 2.2 Method of Solving the Equation

We will now try to solve the linear advection equation (8), using the Lax-Friedrichs scheme. In order to make the handling of the boundary conditions as simple as possible, we assume by forehand that the solution is *periodic* in space, repeating itself on a scale  $2L$ :  $u(x, t) = u(x + 2L, t) = u(x + 2nL, t)$  with  $n \in \mathbb{N}$ . We now can solve the equation in a 'box' of size  $2L$  with boundary condition  $u(-L, t) = u(L, t)$ .

*Fortran* is used to write a program to solve the advection equation (see Appendix). The advection is scaled to unity ( $a = 1$  in equation 8). The CFL condition (equation 14) becomes

$$\Delta \tilde{t} \leq \Delta \tilde{x} \quad (15)$$

We used a grid with 41 evenly spaced points (with  $u_{41}^n = u_1^n$  because of the boundary condition) so that  $\Delta \tilde{x} = 0.05$  and  $\Delta \tilde{t} = 0.8\Delta \tilde{x} = 0.04$ .

The analytical solution of this problem, with boundary conditions  $u(x, 0) = 1$  for  $|x| < 1/3$  and 0 for  $1/3 < |x| \leq 1$ , is given by a 'block' from  $u(x, t) = 1$  for  $(t - 1/3) < x < (t + 1/3)$  and  $u(x, t) = 0$  otherwise. It is a block which doesn't change form and moves with constant velocity  $1 (= \tilde{a})$  to the right.

Using the program and the above values and boundary conditions, the profile of  $u(x, t)$  at time  $\tilde{t} = 4$  is given in Figure 1. At time  $\tilde{t} = 8$  the numerical solution is plotted in Figure 2. They are symmetric around the  $x = 0$  because it moves to the left again when it hits the right boundary. Therefore at each even point in time the function is centered around  $x = 0$

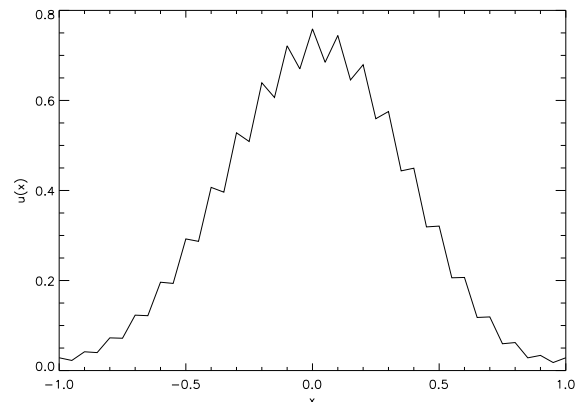


Figure 1: The numerical solution to the advection equation at time  $\tilde{t} = 4$ .

The profiles converge to the analytical profile when  $\Delta \tilde{x}, \Delta \tilde{t} \rightarrow 0$ . As an illustration we show the numerical solution at time  $\tilde{t} = 4$ , with a time step

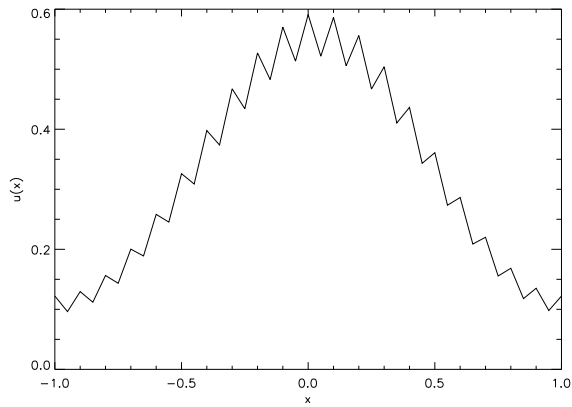


Figure 2: The numerical solution to the advection equation at time  $\tilde{t} = 8$ .

$\Delta\tilde{t} = 0.004$ , 10 times smaller than before, in Figure 3. The shape of this curve is already much closer to the box of the analytical solution, and it is expected that for even smaller time steps the curve will eventually coincide with the theoretical one.

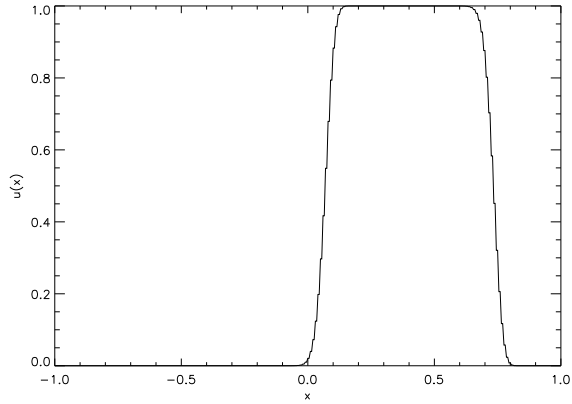


Figure 3: The numerical solution to the advection equation at time  $\tilde{t} = 4$  with a 10 times smaller time step than before.

## Appendix

The code used for this results is presented here. It is made using *Fortran 77*. The plots are made using *IDL*.

```
program integrate
implicit real*8(a-h,o-z) real*8 uoud(41),u(41),x(41)
open(1,name='integrate4.dat',status='unknown') open(3,name='integrate8.dat',status='unknown')
dx=0.05 dt=0.04 t=0 a=0.5*(1-dt/dx) b=0.5*(1+dt/dx)
c ***** analytische oplossing
do j=1,41 x(j)=(j-1.)/20.-1. if(abs(x(j)) .le. 1./3.) then u(j)=1. else u(j)=0. end if enddo
c ***** numerieke oplossing
do n=1,210 t=t+dt
c ***** oude waarden opslaan
do j=1,41 uoud(j)=u(j) enddo
c ***** nieuwe waarden berekenen
u(1)=a*uoud(2)+b*uoud(40) u(41)=u(1) uoud(41)=uoud(1)
do j = 2,39 u(j)=a*uoud(j+1)+b*uoud(j-1) enddo
u(40)=a*uoud(1) + b*uoud(39)
if(n .eq. 100) then do j=1,41 write(1,2) x(j), u(j) enddo endif
if(n .eq. 200) then do j=1,41 write(3,2) x(j), u(j) enddo endif enddo
c ***** einde numerieke oplossing code
2 format(2e16.8)
close(1) close(3)
end
```

In reality, all lines start with six spaces (standard for *Fortran*, except for the command lines (beginning with a c on the first place), and lines which have to do with Format statements (they have a number in those first six places, in this code only 2's).