A Simple Solver for Linear Advection With Constant Coefficients

Mladen Ivkovic (mladen.ivkovic@durham.ac.uk)

Contents

1 Overview

The goal of this exercise is to get you started with debugging using tools such as `gdb` and `valgrind`. To this end, a toy code is provided, both in C and in Fortran, which has been intentionally riddled with bugs for you to find and fix. The buggy code is stored in `exercise/src/c` and `exercise/src/f90`, respectively. Solutions are available in `exercise/solutions/`. You may want to run the codes stored there to compare what the correct output should be.

This overview aims to provide a brief summary of the toy code, its main algorithm, and the equations being solved. Much more information is given in subsequent Sections, but this overview should suffice to get you started.

The toy code solves an equation called "*linear advection with constant coefficients*" in 1D. That equation is given by:

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

with $a = \text{ const } > 0$. An analytical solution exists, and is given by any function $q(x,t)$ which satisfies

$$q(x,t) = q(x - a(t - t_0), t_0). \tag{2}$$

It describes a function $q(x,t)$ of any arbitrary shape being translated by at as time evolves (see e.g. Figure 1). So the solution you should be expecting is the same initial shape traveling along the x-axis with constant speed a, save for some numerical diffusion (see Fig. 4).

To solve the problem numerically, we discretize space in N cells of equal width Δx. The cells are given an integer index i. The numerical solution at a subsequent time step t^{n+1} given the current state t^n is given by:

$$q_i^{n+1} = q_i^n + a\frac{\Delta t}{\Delta x}\left(q_{i-1}^n - q_i^n\right) \tag{3}$$

where the time step size Δt must be limited by the so-called "CFL condition"

$$\Delta t_{max} = C_{cfl} \frac{\Delta x}{a} \tag{4}$$

where $C_{cfl} \in [0, 1)$ is a user-set factor.

For this exercise, we employ *periodic boundary conditions*, i.e. we set up the solver such that whatever leaves or enters on one side of our simulation box also enters or leaves on the opposite side, respectively. To achieve this behavior, we add additional cells ("*ghost cells*") around our simulation box.

Suppose we have 1D grid with N cells and require 2 ghost cells each, which will have indices -2, -1, $N+1$, and $N+2$. We obtain periodic boundary conditions by enforcing

$$\begin{aligned} q_{-2} &= q_{N-1} \\ q_{-1} &= q_N \\ q_{N+1} &= q_0 \\ q_{N+2} &= q_1 \end{aligned}$$

See also Figure 3.

Finally, a brief description of the full algorithm of the solver:

```
To start, set $t_{current} = t^0 = t_{start}$ and set up the initial states $q_i^0$ for each cell $i$.
While $t_{current} < t_{end}$, solve the $n$-th time step:
    Apply the periodic boundary conditions
    Find the maximal permissible time step $\Delta t$ using eq. 4
    For each cell $i$, find the updated states $q_i^{n+1}$ using eq. 3
    Update the current time: $t_{current} = t^{n+1} = t^n + \Delta t$
```

2 Linear Advection With Constant Coefficients

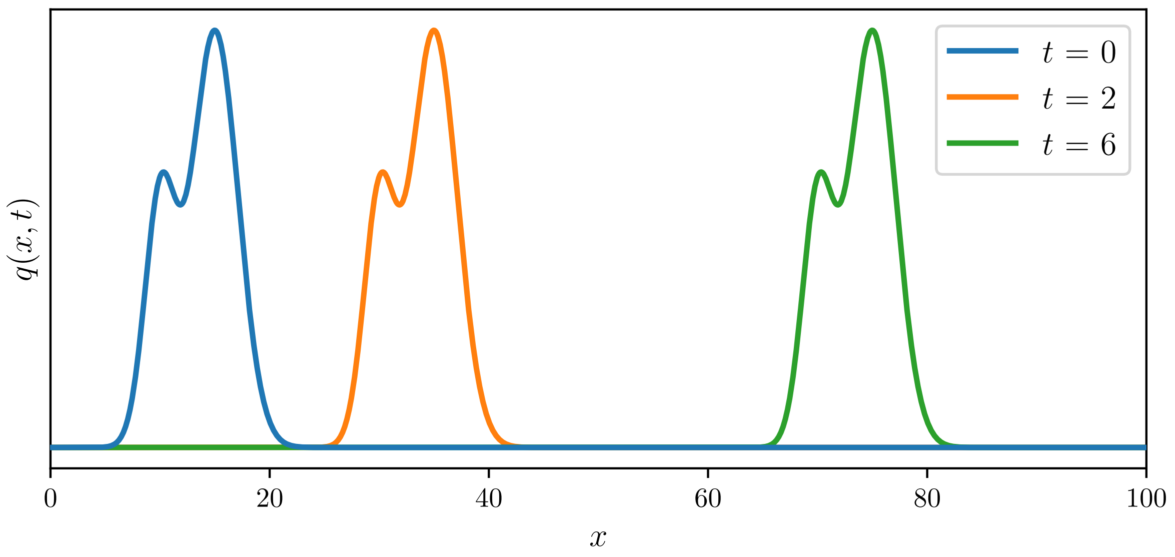

Figure 1: The analytical solution of the linear advection equation with constant coefficients (eq. 8). The solution consists of the same initial shape being translated with at as time evolves.

In principle, we're trying to solve a class of partial differential equations called "Hyperbolic Conservation Laws". They are determined by a "state vector" $\mathcal{U}$ and a "flux tensor" $\mathcal{F}(\mathcal{U})$ (in addition to a couple of other conditions on the relation between $\mathcal{U}$ and $\mathcal{F}(\mathcal{U})$ which don't really matter for this exercise.) Hyperbolic conservation laws take the following form:

$$\frac{\partial \mathcal{U}}{\partial t} + \nabla \cdot \mathcal{F}(\mathcal{U}) = 0 \tag{5}$$

In this exercise, we're focussing on the so-called "linear advection with constant coefficients", which is given by

$$\mathcal{U}(\mathbf{x}, t) = q(\mathbf{x}, t) \tag{6}$$

$$\mathcal{F}(\mathcal{U}(\mathbf{x}, t)) = a \cdot q(\mathbf{x}, t) \quad , \quad a = \text{ const } > 0 \tag{7}$$

effectively giving us in 1D:

$$\boxed{\frac{\partial q}{\partial t} + a \cdot \frac{\partial q}{\partial x} = 0} \tag{8}$$

The analytical solution is given by any function $q(x)$ which satisfies

$$q(x,t) = q(x - a(t - t_0), t_0) \tag{9}$$

This can readily be verified by applying the time derivative (and the following chain rule) to the function 9

$$\frac{\partial}{\partial t} q(x,t) = \frac{\partial}{\partial t} q(x - a(t-t_0), t_0) = \frac{\partial}{\partial x} q(x, t_0) \cdot \frac{\partial}{\partial t}(-a(t-t_0)) = -a \frac{\partial}{\partial x} q(x,t_0) \tag{10}$$

Inserting 10 into 8 instantly shows that the original equation 8 is satisfied.

The solution 9 describes a function $q(x,t)$ of any arbitrary shape being translated by at as time evolves. See Figure 1 for a visualization.

3 Numerical Method

3.1 The Base Update Formula

First things first: We need to discretize the problem. To do so, we use a very simple approach for this exercise, where we replace the analytical derivatives with approximate discrete ones.

But before we can do that, we need to define a discrete version of our initial problem. Suppose you split the domain you want to simulate into N cells of equal sizes Δx. Let each cell have an index i.

We assume that a state q is constant within a cell. See Figure 2 for a visual representation. This approach is called "piecewise constant reconstruction".

Suppose we're solving the equation over some small time step size Δt, and that each cell has the same size Δx. We can now replace the analytical derivatives with discrete

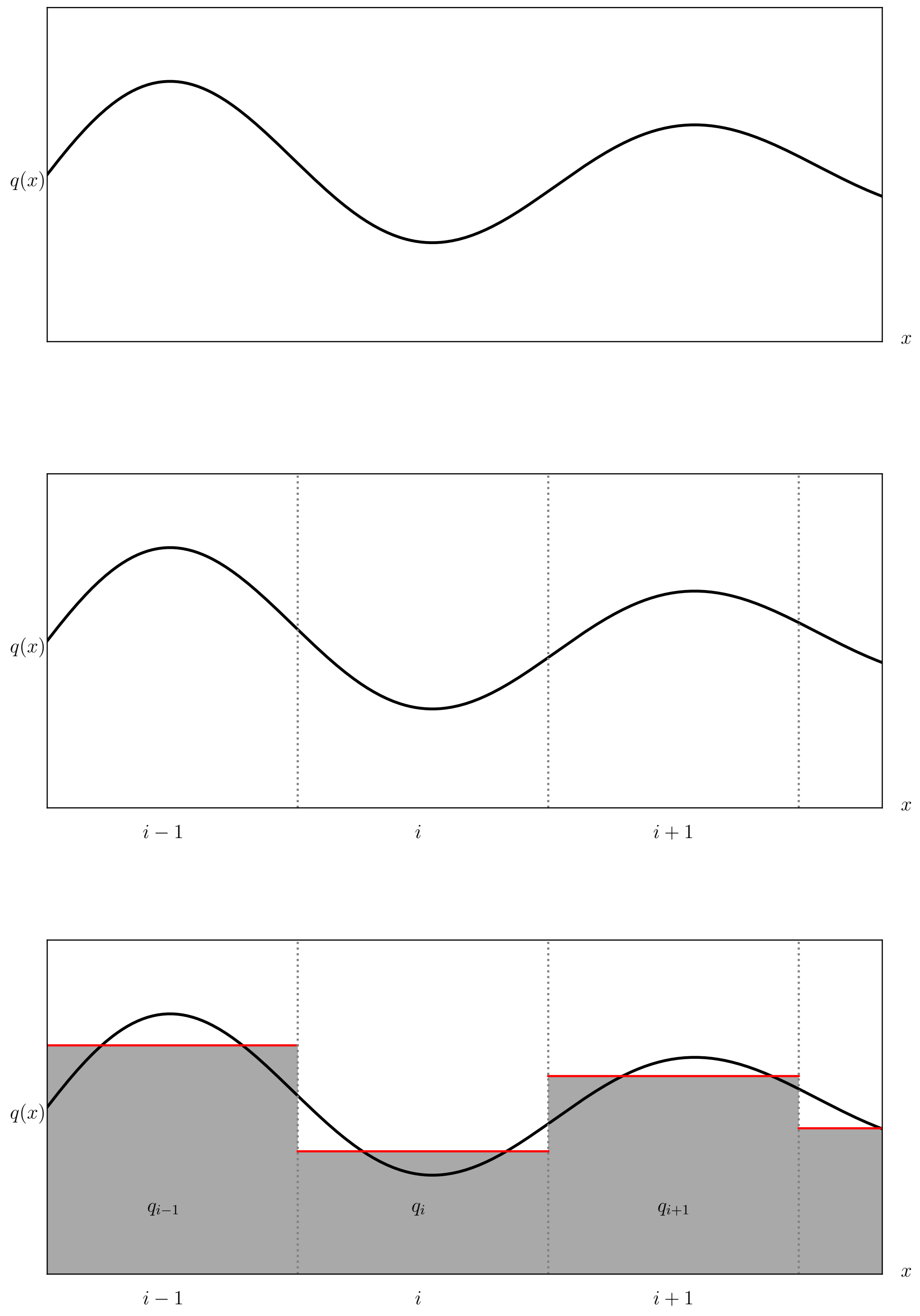

Figure 2: Piecewise constant reconstruction of the field. We start with some function $q(x)$ (top row). Then we introduce cells with equal spacings Δx and indexes $i-1$, i, and $i+1$ (middle row). Finally, we average the initial quantity $q(x)$ inside each cell to obtain that cell's discretized value q_i, assuming that the states within each cell are constant.

ones:

$$\frac{\partial}{\partial t} q(x_i, t) \approx \frac{q(x_i, t + \Delta t) - q(x_i, t)}{\Delta t} \tag{11}$$

$$\frac{\partial}{\partial x} q(x_i, t) \approx \frac{q(x_i, t) - q(x_i - \Delta x, t)}{\Delta x} \tag{12}$$

Using the notation

$$q(x_i) = q_i \tag{13}$$

$$q(x_i - \Delta x) = q_{i-1} \tag{14}$$

$$q(t) = q^n \tag{15}$$

$$q(t + \Delta t) = q^{n+1} \tag{16}$$

we arrive at the update formula:

$$\boxed{q_i^{n+1} = q_i^n + a\frac{\Delta t}{\Delta x}\left(q_{i-1}^n - q_i^n\right)} \tag{17}$$

Note that we also assume that the advection velocity, a, is also constant and positive. If it were negative, we'd need to modify our expression for the approximate gradient in eq. 12.[1]

3.2 The CFL Condition

In our derivation of the numerical method above we simply assumed that the time step size Δt is sufficiently small. But how small is "sufficient"?

The restriction of a maximal time step size is given by the so-called Courant-Friedrichs-

[1]If a were negative, we'd have to modify our gradient as $\frac{\partial}{\partial x} q_i \approx \frac{q_{i+1} - q_i}{\Delta x}$. The important point is that we always do *upwind differencing*. To obtain a finite difference, as we do here, you must never use the value that is downstream, i.e. that is in the direction of the flux, or the flow. Doing that means you'd be taking a value for your computation that won't be valid as soon as an infinitesimal time interval passes, because the ingoing flux will change the downwind state. This is unphysical and leads to violent instabilities.

Levy condition (CFL condition). In principle, that condition is derived through analysis of the numerical method with regards to stability and accuracy. In practice, an intuitive explanation is readily available for the one-dimensional case, which luckily is what we're trying to solve.

Note how in our approximate gradient expression (eq. 12) we only used values of an adjacent cell with index $i-1$. So what would happen if we let the quantity be advected more than a cell width during a single time step, i.e. if the quantity gets advected to two, three, four cells away downstream? The answer is - undefined, unphysical behavior. Our method is not set up to account for changes that occur further away than a single cell width next to it. Therefore, we must limit the time step size such that the quantity never gets advected further than a single cell width during a single time step size. In other words, we must enforce:

$$a \cdot \Delta t_{max} \leq \Delta x \tag{18}$$

This condition is typically written as:

$$\Delta t_{max} = C_{cfl} \frac{\Delta x}{a} \tag{19}$$

with $C_{cfl} \in [0, 1)$ a user-set factor. The lower it is, the more accurate the results, but the more computations (i.e. time steps) you need to do to complete your simulation.

3.3 Boundary Conditions

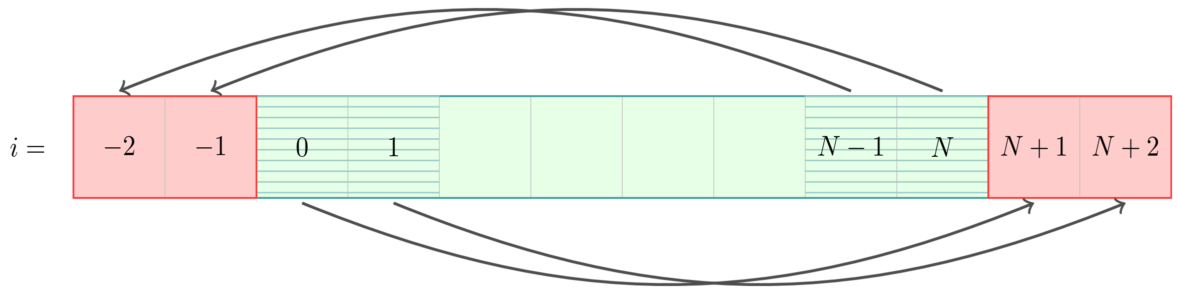

Figure 3: Method to obtain periodic boundary conditions. The ghost cells are red, the arrows show what will be copied where.

For this exercise, we employ *periodic boundary conditions.* Basically this means that we set up the solver such that whatever leaves or enters on one side of our simulation box, enters or leaves on the opposite side, respectively.

To achieve this behavior, we add additional cells ("*ghost cells*") around our simulation box. How many cells you need to add depends on the methods (and mostly stencils) you use. If you only take into account one neighboring cell, then one ghost cell on every boundary suffices. (This is the case that we're working with.) In Figure 3, two such ghost cells for a 1D grid are drawn.

Suppose we have 1D grid with N cells and require 2 ghost cells each, which will have indices -2, -1, $N+1$, and $N+2$. We obtain periodic boundary conditions by enforcing

$$\begin{aligned} q_{-2} &= q_{N-1} \\ q_{-1} &= q_N \\ q_{N+1} &= q_0 \\ q_{N+2} &= q_1 \end{aligned}$$

3.4 The Full Algorithm

Finally, a brief description of the full algorithm of the solver.

```
To start, set $t_{current} = t^0 = t_{start}$ and set up the initial states $q_i^0$ for each cell $i$.
While $t_{current} < t_{end}$, solve the $n$-th time step:
    Find the maximal permissible time step $\Delta t$ using eq. 19
    Apply the periodic boundary conditions
    For each cell $i$, find the updated states $q_i^{n+1}$ using eq. ??
    Update the current time: $t_{current} = t^{n+1} = t^n + \Delta t$
```

4 Results

Figure 4 shows the solution using our first order finite difference method for a step function and a Gaussian at different times, moved back to the original position to demonstrate how the initial shape changes over time. According to the known analytical solution, which says that $q(x, t)$ should only be translated with at as time evolves, this is wrong.

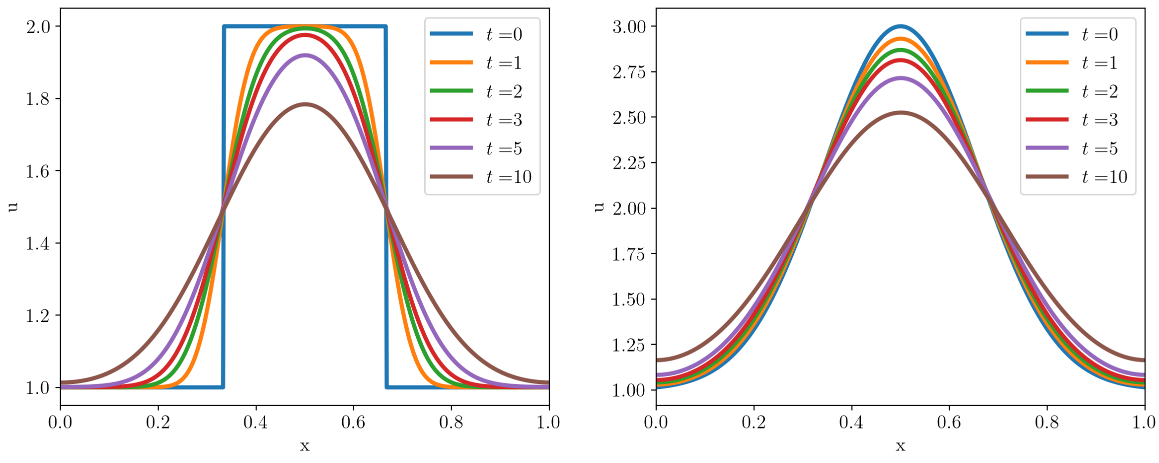

Figure 4: The solution of the linear advection equation with constant coefficient $a = 1$ (in arbitrary units). On the left, the initial conditions are a step function, on the right, they are a Gaussian. The results at $t > 0$ have been repositioned to the original position to demonstrate how the initial shape changes over time, which according to the analytical solution shouldn't happen. The cause of this phenomenon is an effect called "numerical diffusion".

The reason is an effect called "numerical diffusion". It arises due to the nature of the discrete method, and is unfortunately unavoidable.

The purpose of this Section in the text is to give you a known result you can compare yours to, and in particular to mention that the effects of numerical diffusion are natural and expected. You don't need to understand them to successfully debug the solver. However, for the interested readers, an explanation is given below.

Consider our finite difference approximation of the eqation of linear advection:

$$\frac{q_i^{n+1} - q_i^n}{\Delta t} + a\frac{q_i^n - q_{i-1}^n}{\Delta x} = 0 \tag{20}$$

The terms q_i^{n+1} and q_{i-1}^n can be Taylor-expanded to second order in Δt and Δx:

$$q_i^{n+1} = q_i^n + \Delta t\frac{\partial q}{\partial t} + \frac{\Delta t^2}{2}\frac{\partial^2 q}{\partial t^2} + \mathcal{O}(\Delta t^3) \tag{21}$$

$$q_{i-1}^n = q_i^n - \Delta x\frac{\partial q}{\partial x} + \frac{\Delta x^2}{2}\frac{\partial^2 q}{\partial x^2} + \mathcal{O}(\Delta x^3) \tag{22}$$

Inserting these expansions into the discretized equation 20 gives

$$\begin{aligned}
&\frac{1}{\Delta t}\left[q_i^n + \Delta t\frac{\partial q}{\partial t} + \frac{\Delta t^2}{2}\frac{\partial^2 q}{\partial t^2} + \mathcal{O}(\Delta t^3) - q_i^n\right] + \\
&\frac{a}{\Delta x}\left[q_i^n - q_i^n + \Delta x\frac{\partial q}{\partial x} - \frac{\Delta x^2}{2}\frac{\partial^2 q}{\partial x^2} + \mathcal{O}(\Delta x^3)\right] = 0 \qquad (23)\\
=&\frac{\partial q}{\partial t} + \frac{\Delta t}{2}\frac{\partial^2 q}{\partial t^2} + a\frac{\partial q}{\partial x} - a\frac{\Delta x}{2}\frac{\partial^2 q}{\partial x^2} + \mathcal{O}(\Delta t^2) + \mathcal{O}(\Delta x^2) \qquad (24)
\end{aligned}$$

keeping only the first order terms in Δt and Δx, this can be rearranged to

$$\frac{\partial q}{\partial t} + a\frac{\partial q}{\partial x} = a\frac{\Delta x}{2}\frac{\partial^2 q}{\partial x^2} - \frac{\Delta t}{2}\frac{\partial^2 q}{\partial t^2} \tag{25}$$

The left hand side of eq. 25 is identical to the conservation law we are solving, and should be zero. Hence we can interpret everything on the right hand side as the highest order error term that the numerical scheme introduces. Let's name it *Err*:

$$Err = a\frac{\Delta x}{2}\frac{\partial^2 q}{\partial x^2} - \frac{\Delta t}{2}\frac{\partial^2 q}{\partial t^2} \tag{26}$$

To proceed, we express $\frac{\partial^2 q}{\partial t^2}$ as a function of $\frac{\partial^2 q}{\partial x^2}$ by differentiating the analytical advection equation once w.r.t. t:

$$\frac{\partial}{\partial t}\left(\frac{\partial}{\partial t}q + a\frac{\partial}{\partial x}q\right) = \frac{\partial^2 q}{\partial t^2} + a\frac{\partial^2 q}{\partial x \partial t} = 0 \tag{27}$$

and once w.r.t. x:

$$\frac{\partial}{\partial x}\left(\frac{\partial}{\partial t}q + a\frac{\partial}{\partial x}q\right) = \frac{\partial^2 q}{\partial x \partial t} + a\frac{\partial^2 q}{\partial x^2} = 0 \tag{28}$$

Relating these two derivatives over their common term $\frac{\partial^2 q}{\partial x \partial t}$ gives us the required relation:

$$-a\frac{\partial^2 q}{\partial x \partial t} = \frac{\partial^2 q}{\partial t^2} = a^2\frac{\partial^2 q}{\partial x^2} \tag{29}$$

Which allows us to express the error Err as

$$\begin{align} Err = a\frac{\Delta x}{2}\frac{\partial^2 q}{\partial x^2} - \frac{\Delta t}{2}\frac{\partial^2 q}{\partial t^2} &= a\frac{\Delta x}{2}\frac{\partial^2 q}{\partial x^2} + a^2\frac{\Delta t}{2}\frac{\partial^2 q}{\partial x^2} \tag{30}\\ &= \frac{a\Delta x}{2}\left(1 - \frac{a\Delta t}{\Delta x}\right)\frac{\partial^2 q}{\partial x^2} \tag{31}\\ &= \frac{a\Delta x}{2}\left(1 - C_{CFL}\right)\frac{\partial^2 q}{\partial x^2} \tag{32}\end{align}$$

Comparing this result and eqns. 25 and 26 with the advection-diffusion equation[2]

$$\frac{\partial}{\partial t}q + a\frac{\partial}{\partial x}q = D\frac{\partial^2 q}{\partial x^2} \tag{33}$$

it is clear that the error term that is introduced by the discretization of the equations is in fact a diffusion term with the diffusion coefficient $D = \frac{a\Delta x}{2}\left(1 - C_{CFL}\right)$. This expression for D also tells us how the diffusivity of the method will behave:

- $D \propto \Delta x$: The diffusivity decreases with smaller grid spacing Δx
- $D \propto (1 - C_{CFL})$: The diffusivity decreases with bigger (maximal) time step sizes C_{CFL}.

[2]also called the "convection-diffusion equation"