# Lecture 9

Continuum techniques: Flux limiters

D. Dickinson
d.dickinson@york.ac.uk

York Plasma Institute, School of PET, University of York

Semester 1

## Overview of course

This course provides a brief overview of concepts relating to numerical methods for solving differential equations.
Topics to be covered include:

- ~~Integrating ODEs~~
  - ~~Explicit techniques~~
  - ~~Implicit techniques~~
  - ~~Using Scipy to integrate ODEs~~
- ~~Spatial discretisation~~
  - ~~Finite differencing~~
  - ~~Spectral methods~~
  - ~~Finite elements~~
- ~~Particle In Cell (PIC) approaches~~
- Continuum techniques

flux limiters

Notes available on the VLE.

## Overview this lecture

This lecture will look at

- An introduction to grid based methods

## An alternative to the PIC approach

In the previous lecture we looked at the PIC approach, where we model the distribution function using a set of particles/markers which we allow to move around the computational domain. This is a relatively simple approach that brings several benefits, but we also discovered that noise is often a concern in such simulations and it is difficult to improve on this.

solely grid-based

Today we will look at an alternative approach known as grid based or continuum techniques. Rather than tracking a random sample of particles, the continuum approach evolves the distribution function[1] itself on a discrete grid. This is typically more complicated than the PIC approach, but doesn't suffer from noise and it can be easier to conserve quantities like energy exactly[2].

We'll see that we generally need to introduce special techniques to keep these continuum approaches stable whilst minimising the introduction of artificial features such as diffusion, dispersion and spurious oscillations.

[1]The distribution function will typically be a function of both space and velocity.
[2]This can be quite important for nonlinear simulations.

## An example problem

We'll use a simple example throughout this lecture to help illustrate different concepts:

$$\frac{\partial f}{\partial t} = -v\frac{\partial f}{\partial x}$$

describes a flow of velocity v (assume constant)

which is the 1D advection problem (hyperbolic).

we need some way of calculating derivative with x and then integrate with time

We can use our knowledge of finite differences and explicit time integration to discretise the equation to yield

explicit euler method for forward integration

$$\frac{f_j^{i+1} - f_j^i}{\delta t} = -v\frac{f_{j+1}^i - f_{j-1}^i}{2\delta x}$$

central difference for derivative

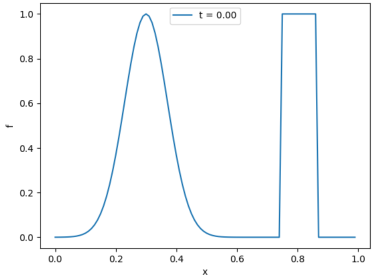
where we have used a 2nd order central difference for the $x$ derivative and the 1st order explicit approximation for the $t$ derivative. Here superscripts represent the time step and subscripts are used for the spatial grid index.

## An example problem

Rearranging the previous expression we can obtain an equation for the values at the next time point

$$f_j^{i+1} = f_j^i - v\frac{\delta t}{2\delta x}\left(f_{j+1}^i - f_{j-1}^i\right)$$

Given initial conditions we can use this to evolve $f\left(x\right)$ in time.



Integrate from 0 to t=1 and compare initial and final state - we want them to be the same

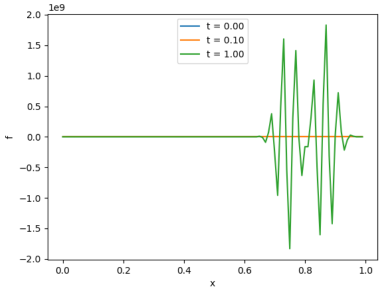
Figure: State at $t = 0$ showing smooth Gaussian plus a top-hat function.
good for tests

# An example problem

Rearranging the previous expression we can obtain an equation for the values at the next time point

$$f_j^{i+1} = f_j^i - v\frac{\delta t}{2\delta x}\left(f_{j+1}^i - f_{j-1}^i\right)$$

Given initial conditions we can use this to evolve $f\left(x\right)$ in time. Implementing this for a periodic domain in python (see eulercentral.py) and running gives



Gone wrong.

Figure: State at $t = 0, 0.1, 1.0$ showing large numerical instability developing by $t = 1$.

## An example problem : Stability analysis

We can see that things aren't behaving nicely with this simple approach, indeed this method is unstable. To understand why let's consider introducing a single Fourier harmonic

$$f_j^i = \underline{\hat{f}^i} \exp\left(ikx_j\right)$$

how big is it at this time step

and

$$f_{j\pm1}^i = \hat{f}^i \exp\left(ik\left[x_j \pm \delta x\right]\right) = f_j^i \exp\left(\pm ik\delta x\right)$$

all that changes is x position

Substituting this into our combined Euler-central expression we find

$$\hat{f}_j^{i+1} = \hat{f}_j^i - v\frac{\delta t}{2\delta x}\hat{f}_j^i\left[\exp\left(ik\delta x\right) - \exp\left(-ik\delta x\right)\right]$$

allows us to pull out amplitude
at current location (f_hat_ji)
and we can divide through

This allows us to calculate the amplification factor, $r$

$$r = \frac{\hat{f}^{i+1}}{\hat{f}^i} = 1 - iv\frac{\delta t}{\delta x}\sin\left(k\delta x\right)$$

in general expect r<1
(function not growing)

need right term to be 0

take magnitude so right term squared, so is plus that term.

## An example problem : Stability analysis

need to find a different way.
don't use explicit euler!!!

We can see that $|r| > 1$ for any $k \neq 0$, meaning this approach in unconditionally unstable. We require $|r| \leq 1$ to be stable in this case. In the situation where we expect a growing solution then we instead require $|r| \leq 1 + \mathcal{O}(\delta t)$ for stability.

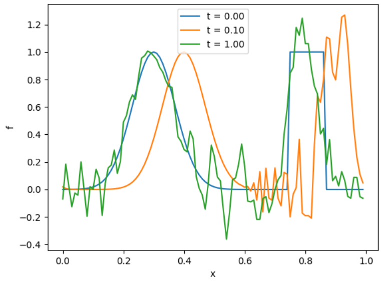
This approach to analysing the stability of a numerical scheme is known as von Neumann stability analysis. Whilst it strictly only applies to linear problems it is often applied to non-linear cases with some margin for error.

can use to see if the approach is going to work or not

## An example problem : Avoiding instability

We can improve on our inital scheme by replacing the Euler time integration with SciPy's solve_ivp. This is implemented in central.py

in plasmas can have shocks - sharp changes



taylor expansion requires it to be continuously differentiable

Figure: State at $t = 0, 0.1, 1.0$ showing much better agreement than before at $t = 1$ but artificial oscillations.

where the initial conditions (blue) include a smooth Gaussian and a top hat region. We can see that the result at $t = 1$, which should be identical to the initial conditions, is somewhat oscillatory. It matches the Gaussian section reasonably well but is fairly poor for the top hat region. This is effectively due to the break down of the Taylor expansion in regions of steep gradient/discontinuity. sharp change

## An example problem : Upwinding

So how can we improve on the previous approach? We can actually return to our original Euler+Central scheme and make a small modification

$$f_j^{i+1} = f_j^i - v\frac{\delta t}{\delta x}\begin{cases} f_j^i - f_{j-1}^i & v > 0 \\ f_{j+1}^i - f_j^i & \text{otherwise} \end{cases}$$

pick which side we look at
depending on sign of velocity
- tells us which derivative (gradient)
the stuff flowing towards us has

less accurate, need more grid points

This is known as (1st order) upwind differencing and we've replaced our 2nd order central difference with 1st order one sided differences, where the differencing direction is picked to match the characteristic flow.

The von Neumann stability analysis for this scheme gives $|r| \leq 1$ provided

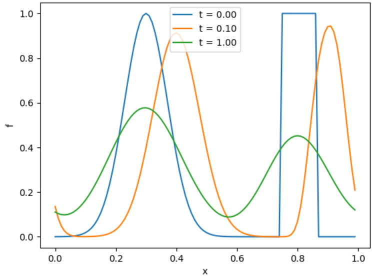
$$v\frac{\delta t}{\delta x} < 1$$

can't flow more than one grid
space in one time step

One may interpret this as saying the flow in a single timestep cannot take data more than one grid space. This is the Courant-Friedrichs-Lewy (CFL) limit. Unfortunately this refers to *any* information flowing across the grid, so your timestep is limited by the shortest time scale in your system (e.g. light waves!). This limitation can be avoided with implicit methods.

## An example problem : Upwinding

We can implement this upwind difference for our test problem (see upwind.py) and, using an appropriate timestep, we find



A bit better, neater, but still not perfect - amplitudes not great and tophat not square

- smoothed out original function

Figure: State at $t = 0, 0.1, 1.0$ showing heavily smoothed structure and strong damping by $t = 1$.

There are no spurious oscillations and the solution remains positive, but there is quite poor agreement between the t=0 and t=1 results. It looks like the curve has been heavily smoothed/damped, and *indeed it has*!

## An example problem : Numerical diffusion

Solution not perfect

The spreading of the solution is because the 1st order differences we've introduced have a second order error term that looks like

No control over it, error

$$\epsilon \sim \frac{d^2 f}{dx^2}$$

describes diffusion (things spreading out)
- damps out oscillations

which looks like a diffusion term. This numerical diffusion damps oscillations and stabilises the method but leads to an unphysical level of diffusion. This leads to a requirement for a very high grid resolution to minimise the numerical diffusion[3].

How can we avoid this numerical diffusion? The Lax-Wendroff scheme returns to our original (unstable) central difference approach and adds in some diffusion.

euler for time, central difference for spatial differencing, add in extra term for diffusion

$$f_j^{i+1} = f_j^i - v\frac{\delta t}{2\delta x}\left(f_{j+1}^i - f_{j-1}^i\right) + \frac{1}{2}\left(\frac{v\delta t}{\delta x}\right)^2\left(f_{j+1}^i - 2f_j^i + f_{j-1}^i\right)$$

disappears in the limits. 2nd order central difference.

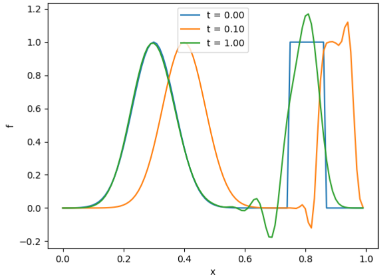
[3] The reader should try running the example with different nx.

## An example problem : Numerical diffusion

This can also be written as the upwind method plus some "anti-diffusion"

$$f_j^{i+1} = f_j^i - s\left(f_j^i - f_{j-1}^i\right) - \frac{s\left(s-1\right)}{2}\left(f_{j+1}^i - 2f_j^i + f_{j-1}^i\right)$$

where we've used $s = v\delta t/\delta x$ and considered $v > 0$. The anti-diffusion cancels some of the errors in the 1st order upwind method leading to (see lax_wendroff.py)



better, but still not perfect.

Figure: State at $t = 0, 0.1, 1.0$ showing very good agreement where solution is smooth but oscillations near sharp gradients.

## An example problem : Spurious oscillations

With the Lax-Wendroff scheme the numerical diffusion is small and the smooth regions are very well described. There are however some problems with the top hat region, namely that over (and under) shoot oscillations have been introduced near the discontinuities. The source of this is the third order error term, which leads to different wavelengths having slightly different velocities. This is known as numerical dispersion. Structures will spread out in a way that distorts them, rather than just smoothing them out.

For problems where the solution is known to be smooth or small overshoots are not important such schemes can work very well. However, in some situations these overshoots are not acceptable as they can lead to unphysical situations (like negative density). Depends on the problem if you want to fix this or not.

## An example problem : Spurious oscillations

So how do we fix this? For numerical diffusion we managed to solve the issue by going to the next order, so why don't we do that here?

Unfortunately Godunov's theorem states that

**Godunov's theorem**

Linear numerical schemes for solving PDE's, having the property of not generating new extrema, can be at most first-order accurate.

This suggests that we either have to accept large numerical diffusion or overshoot oscillations, neither of which are particularly good options.

So what can we do? Prevent our scheme from being linear so Godunov's theorem no longer applies!

to use more than first order

If we can detect where there is sharp gradients, use different method

## An example problem : Adaptive schemes

If we use different schemes in different regions we can get away from the limitations of Godunov's theorem. These are adapative methods which use high-order approximations in smooth regions and revert to first order approaches when near discontinuities (or sharp gradients).

A common technique to achieve this is to limit gradients/fluxes such that new maxima and minima are not created. These techniques are referred to as limiter methods[4].

Look at the derivative on left and right. If we have smooth function, these derivatives should be roughly the same, if sharp then they will be very different; then pick between lower or higher order methods.

[4] Also known as flux limiters or slope limiters.

# An example problem : Flux limiters

We can write our test problem in terms of fluxes, $F$, between cells [5]

G - gradient
F - flux

$$\frac{\partial f_j}{\partial t} + \frac{1}{\delta x}\left[F\left(f_{j+1/2}\right) - F\left(f_{j-1/2}\right)\right] = 0$$

Now we can write our flux function, $F$, in terms of values calculated using both low and high order schemes

$$F\left(f_{j+1/2}\right) = G^{\mathrm{l}}_{j+1/2} - \phi\left(\theta_j\right)\left(G^{\mathrm{l}}_{j+1/2} - G^{\mathrm{h}}_{j+1/2}\right)$$

Phi is a function not defined here, there are different potential choices

where $G^{\mathrm{l}}_{j+1/2}$ and $G^{\mathrm{h}}_{j+1/2}$ are the low and higher order flux values and $\phi\left(\theta_j\right)$ is known as the flux limiter function and

$$\theta_j = \frac{f_j - f_{j-1}}{f_{j+1} - f_j}$$

looking backwards

looking forwards

theta is difference between function at left and right
- 1 if smooth
- far from 1 if sharp

is the ratio of the neighbouring gradient estimates.

So if theta is 1, we just end up with high order gradient

[5] This is often related to something known as the finite volume method/approach; we consider a small volume around each grid point and treat flow into and out of this volume. As what flows out of one volume must flow into another these approaches are known as conservative.

## An example problem : Flux limiters

There are a wide range of flux limiter functions, $\phi$, but all satisfy $\phi(\theta) \geq 0$. When the function is 0 we fall back to our low order scheme suitable for steep gradient regions.

No one limiter is the best choice in all situations and it's usually best to try out different options for the particular problem at hand. Indeed this is still an active area of research and efforts are ongoing to find alternatives which avoid some of the shortcomings of simple limiter techniques. The interested may wish to look in more detail at MUSCL, ENO/WENO and the piecewise parabolic method (PPM) for more information.

## An example problem : Flux limiters

Here we'll just consider the van Leer limiter[6].

just a choice of limiter

$$\phi(\theta) = \frac{\theta + |\theta|}{1 + |\theta|}$$

This is implemented in limiter.py (along with other options) and gives

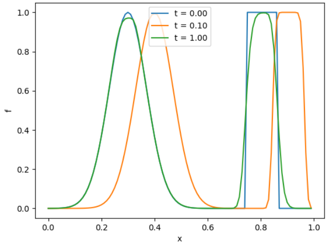


Figure: State at $t = 0, 0.1, 1.0$ using van Leer flux limiter. Smooth region well reproduced and no oscillations near sharp gradients.

This is a pretty good approximation of the solution with no additional oscillations and minimal diffusion. There is some disagreement near the peaks and more modern methods look to fix this.

choosing a different limiter or using more advanced techniques

[6] The reader should refer to other sources such as wikipedia for other options.