Numerical Stability & Advection
Exercise
Dresden Summer School in Systems Biology 2018

Stability of time discretisation schemes

You are familiar with methods to find *analytical* solutions for an ordinary differential equation (ODE), i.e. differential equations of functions in one variable. In the following, we want to take such a simple ODE to understand the basic idea of numerical stability.

a) Give a short definition of the terms (linear and non-linear) ordinary differential equation and partial differential equation.

b) Determine the *analytical* solution $c(t)$ for the following ODE

$$\frac{d}{dt}c(t) = -\lambda c(t)$$

for some constant $\lambda > 0$ and initial condition $c(t = 0) = c_0$. If you did not come across this equation before, read about a method to solve ODEs, which is called separation of variables and apply it here. This equation could describe the continuous degradation of some chemical c with a rate λ.

c) Implement a simple Euler method in your favourite programming language and assume $\lambda = 1$. Analyze your numerical solutions for three different time steps:

- Set $\Delta t = 0.1$. Your numerical solution should be very close to the analytical solution for all t. Your discretization scheme is stable.
- Set $\Delta t = 1.8$. Observe your numerical solution for small t. You should find an oscillatory, but bounded behaviour. Your discretization scheme is marginally stable.
- Set $\Delta t = 5$. Observe the behavior of your numerical solution for larger t. You should find that it is not bounded any more (it 'blows up'). Your discretization scheme is unstable.
- By playing around with Δt, can you find the values where one transits from the stable to the marginally stable and from there into the unstable regime and how this depends on λ? Can you maybe even derive those values from your discretization scheme?

(Optional) In order to evolve an ODE over long times, it is desirable to use large time steps and sometimes an Euler scheme is does not suffice for this purpose. One way out is to use 'higher order' schemes (What does this mean? Read about the idea of Runge-Kutta methods!). Find and implement the second order Runge-Kutta scheme (midpoint-method) for the above ODE and compare its stability properties to the Euler integration.

Advection

Numerical (in-)stability comes in many different forms (e.g. oscillations as discretization artifacts and/or non-boundedness) and does not only depend on the chosen discretization schemes but also on the nature of the underlying differential equation. In the following this will be demonstrated by using the advection equation, which appears often in models that describe flows and transport of chemical substances.

e) Read about the Eulerian and Lagrangian descriptions of a flow field. Explain the key difference in your own words.

Eulerian description

f) Consider the one-dimensional advection equation in an Eulerian description

$$\partial_t c(x,t) + u\partial_x c(x,t) = 0,$$

where $u = const.$ is a constant flow field, which transports for example molecules that are present at a point x and time t at a local concentration $c(x,t)$. Show that the function $c_0(x-ut)$ satisfies the advection equation for an initial condition $c_0(x)$ at $t=0$ and therefore is a solution. Convince yourself that this represents a translation of $c_0(x)$ with velocity u.

g) Consider the advection equation from above on an interval $x \in [-1,1)$ with periodic boundary conditions and initial condition $c_0(x) = \cos^2(x\pi/2)$. We suggest two discretisation schemes for the spatial derivative on a regular grid with grid point spacing h and a time stepping Δt in each iteration. The grid points are therefore located at $x_i = x_0 + ih$ and the times at which the numerical solution is calculated are $t_n = n\Delta t$. To simplify the notation we write $c_i^n := c(x_i, t_n)$. With this, we can express the two discretisation schemes as

$$\frac{\partial c(x_i,t_n)}{\partial t} = -u\frac{c_i^n - c_{i-1}^n}{h} \qquad \text{upwind}$$
$$\frac{\partial c(x_i,t_n)}{\partial t} = -u\frac{c_{i+1}^n - c_{i-1}^n}{2h} \qquad \text{FTCS.}$$

- Discretize the time derivative again using the Euler method, as done in c).
- Rearrange the resulting schemes into the explicit form $c_i^{n+1} = f(c_i^n, c_{i-1}^n, c_{i+1}^n)$ (find the function f), i.e. express the concentration values at time t_{n+1} as a function of the concentration values at previous time t_n.
- Implement the two schemes in your favourite programming language.

In the following, we want to investigate the stability properties of those two schemes. For this purpose, fix $u = 0.2$ and $h = 0.02$.

- For the upwind scheme try out $\Delta t_a = 0.007$ and $\Delta t_b = 0.2$. What do you observe in each case? This scheme is conditionally stable. (keyword *CFL condition*)
- To achieve a higher resolution, reduce the grid point distance to $h = 0.01$. What happens now, if you use time steps of size Δt_a or Δt_b and what does this imply in general for this method?
- Can you find any value of Δt, for which FTCS is stable? You should not, because this scheme is in fact unconditionally unstable.

Note that in principle also here we could have chosen other time steppings to change and potentially improve the properties of the schemes, as discussed before. You may read more about the formal stability analysis of these and other schemes here:
http://www.aei.mpg.de/~rezzolla/lnotes/Evolution_Pdes/evolution_pdes_lnotes.pdf, Chapter 3, or from any other source you find suitable.

Lagrangian description and Cellular Automata

h) In a Lagrangian description, a point at x_i is initially assigned the concentration value $c_0(x_i)$. Each point keeps this initially assigned concentration, while it is advected with velocity u along the domain. Implement such a scheme to understand if there are any time step limitation for a constant advection field u.

i) There are various ways to implement advection in terms of cellular automata (CA). What condition needs to be fulfilled for the fraction $u\Delta t/h$ (to find this condition, remember that the cells of a CA have fixed positions), such that your Lagrangian implementation from question h) becomes an exact CA simulation of the advection problem? If this condition on $u\Delta t/h$ is not satisfied, what rule could one define instead (there are many possibilities) to approximate the advection problem using CA?

j) So far the flow field u was constant ($\partial_x u = 0$) but in general u can and often does vary in space, so that $\partial_x u \neq 0$. Simulating the advection of molecules by such an inhomogeneous flow field requires a modification of your Lagrangian model implemented in h). Read about the continuity equation and compare its form for $\partial_x u = 0$ and $\partial_x u \neq 0$ to find out, how you have to modify your implementation. You should find that the initially assigned concentration on each particle changes in each time step.

The Lagrangian description of an advection problem has an important property, which makes it often a more powerful tool than discretisation schemes as in question g). You have seen that if you decrease h (increase the resolution) in the upwind scheme, you also have to reduce the time step size Δt, in order to have a stable scheme. In fact, for $h \to 0$ also Δt must go to zero so that the upwind scheme becomes useless. In a Lagrangian formulation, one can in principle increase the resolution arbitrarily and still simulate with a finite-time step. In the following, we want to understand this in more detail.

k) The time step constraint for a Lagrangian discretization comes from the requirement, that particles (1D) or streamlines (2D) must not cross within an advection time step. Derive the condition that this implies on Δt in 1D and check if the result is consistent with your answer to h). Make an educated guess how this generalizes to 2D. As a test case, you may implement the 1D advection equation in a Lagrangian description with $u = \cos^2(x\pi/2)$.