Problem Set 9

Exercises for the lecture Fundamentals of Simulation Methods, WS 2020

Prof. Dr. Frauke Gräter, Prof. Dr. Friedrich Röpke
Tutors: Benedikt Rennekamp (benedikt.rennekamp@h-its.org), Fabian Kutzki (fabian.kutzki@h-its.org), Giovanni Leidi (giovanni.leidi@h-its.org), Siddhant Deshmukh (sdeshmukh@lsw.uni-heidelberg.de)
Offices: Heidelberg Institute for Theoretical Studies, Schloss-Wolfsbrunnenweg 35

Hand in until Wednesday, 27.01, 23:59
Tutorials on Friday, 29.01
(Group 1, Giovanni Leidi 09:15)
(Group 2, Benedikt Rennekamp 09:15)
(Group 3, Siddhant Deshmukh 11:15)
(Group 4, Fabian Kutzki 14:15)

1. Numerical hydrodynamics – part 1

1.1. Advection in 1D [12 pt.]

To get ready for solving general hydrodynamics problems in 1D, let us visit the simple advection problem. Consider a 1D domain in the coordinate x, ranging from $x = -L/2$ to $x = +L/2$ for some value of L. In this exercise we take $L = 10$, but your program should keep this a free variable. Let us investigate the time evolution of a function $q(x,t)$ within this domain. We require $q(x,t)$ to obey the advection equation

$$\frac{\partial q(x,t)}{\partial t} + v\frac{\partial q(x,t)}{\partial x} = 0\,, \tag{1}$$

where $v > 0$ is a constant. Let us take $v = 1$. For this simple problem we know the analytic solution: $q(x, t > 0) = q(x - vt, t = 0)$, where we impose the periodic boundary condition implicitly.

Let us new try to solve this equation numerically. We set up a grid in x with $N = 100$ equally spaced grid points between $x = -L/2$ and $x = +L/2$ and with Δx being the cell size. Furthermore, we add one ghost cell on each side to make the implementation of the boundary conditions easier. In total we thus have 102 grid points in x. As initial condition we set

$$q(x,0) = \begin{cases} 1 & \text{for} \quad x < 0\,, \\ 0 & \text{for} \quad x \geq 0\,. \end{cases} \tag{2}$$

As boundary condition we impose $q(-L/2,t) = 1$ and $q(L/2,t) = 0$, which is enforced simply by (re-)setting the values of q in the ghost cells to the boundary value after each time step. Integrate in time from $t = 0$ to $t = 3$ using 100 equal-size time steps Δt.

1. Write a program to perform this numerical integration using the *symmetric* numerical derivative operator $(q_{i+1} - q_{i-1})/2\Delta x$. Show that this scheme will lead to a numerical instability.

2. Now use the one-sided numerical derivative operator $(q_i - q_{i-1})/\Delta x$. Show that this so called *upwind* scheme stays stable.

3. Now use the one-sided numerical derivative operator $(q_{i+1} - q_i)/\Delta x$. Show that this approach is unstable again.

A good way to visualize the stability properties of the different numerical methods is to plot the values of $q(x,t)$ across the entire computational domain at different times. This exercise demonstrates that only the upwind advection algorithm leads to a stable solution.

Let us investigate this scheme further, and from now on only use the upwind method.

4. Put the left boundary condition to $q(-L/2,t) = 0.5$. What happens, and why?

5. Reset, and put the right boundary condition to $q(+L/2,t) = 0.5$. What happens, and why?

6. Reset again, and use a $10\times$ smaller time step, i.e. take 1000 time steps between $t = 0$ and $t = 3$. What is the difference in the result at $t = 3$? Does it become better or worse?

7. Repeat, but now use a $10\times$ *larger* time step, i.e. take 10 time steps between $t = 0$ and $t = 3$. Explain what happens.

1.2. General-purpose 1D advection subroutine/function [8 pt.]

Now let us create a general-purpose computer function for the purpose of 1D numerical advection of any given 1D function $q(x,t)$ (represented as a 1D array q_i^n) for any given velocity (also represented as a 1D array $v_{i+1/2}$). The function should receive the current values of q_i^n, the velocities $v_{i+1/2}$ and a time step size Δt. It should return the values q_i^{n+1}, i.e. the values of q at the next time step. For example, the function call in Python could look like `qnew = advect(qold,vold,dx,dt)`, which advects q_i^n one time step. Use ghost cells to implement the boundary conditions.

You will use this function next week to create a general 1D hydrodynamics code from that, so please test this function and make sure that it works correctly.

Note that the velocities $v_{i+1/2}$ are located *in between* the grid points. In the picture of grid cells: the velocities $v_{i+1/2}$ are defined on the *cell boundaries* while the to-be-advected function values q_i^n are located in the *cell centers*. That also means that the length of the velocity array differs by one from the length of the q array.

8. Apply your function to exercise 1.1 above (with a constant velocity $v = 1$) and convince yourself that your function reproduces the same results.

9. Now let us choose the velocity $v(x) = -2x/L$. This is a converging velocity field. Take as an initial density profile $q(x,0) = 1$ for $|x| \leq L/4$ and $q(x,0) = 0$ for $|x| > L/4$. Integrate again in time from $t = 0$ to $t = 3$ with 100 time steps. Plot the result at different times and explain the result.