Exercises for Numerical Fluid Mechanics (WS2012/13)

Volker Springel & Cornelis Dullemond

Exercise sheet 3 *(duration: 2 weeks)*

A first hydrodynamics solver

We now develop a robust advection subroutine that we will use for making a simple 1-D hydrodynamics program.

1. **Improving the advection method**
   In the previous exercise you have created a program for advecting some function over a grid. Now generalize this on several fronts[1]:

   - Introduce *ghost cells*: 1 ghost cell on each boundary. This means that $x[0]$ is the ghost cell left and $x[N+1]$ is the ghost cell right. In total we thus have $N+2$ cells. Make sure to update the cell values only for the non-ghost cells.
   - Allow to specify a x-dependent velocity field specified *at the cell interfaces*, i.e. $u_{i-1/2}$. Please choose the indexing of the velocity array such that $u[i] = u_{i-1/2}$, i.e. interface i (= actually interface $i-1/2$) is in between cells $i-1$ and i.
   - Use the donor-cell algorithm for the advection. Please make sure that the algorithm works for any sign of u (which may be different at different interfaces).
   - *(Voluntary)* Include the possibility to use the MINMOD and SUPERBEE flux limiters (or slope limiters, which is equivalent in this case).

2. **All-purpose advection subroutine**
   Make a subroutine `advect(n,dx,q,qnew,ui,dt)` where the arguments are[2]:

   `n` Number of grid points N (excluding the ghost cells).

   `dx` The (fixed) grid cell size.

   `q` Array of values q_i^n (array of $N+2$ values).

   `qnew` Array of new values q_i^{n+1} (array of $N+2$ values), i.e. the result of this subroutine.

   `ui` Array of values $u_{i-1/2}$ (array of $N+3$ values).

   `dt` Time step.

   This subroutine returns the `qnew` array, which contains the values of q_i at the new time step.
   Please test your subroutine well, before going on to the next assignment.

[1] You might want to check out the next assignment (putting this all in a subroutine) and do this simultaneously.

[2] You have, of course, the freedom to design your subroutine differently if that is convenient; this is meant merely as an example.

3. **An isothermal 1-D hydrodynamics solver**

Isothermal hydrodynamics in 1-D is given by the following set of PDEs:

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u)}{\partial x} = 0 \tag{13}$$

$$\frac{\partial(\rho u)}{\partial t} + \frac{\partial(\rho u^2)}{\partial x} = -\frac{\partial(\rho c_s^2)}{\partial x} \tag{14}$$

with c_s^2 the isothermal sound speed which is taken to be a constant.

(a) Use the above advection routine for the simple classic scheme described in the lecture notes to solve this set of equations.

(b) Implement periodic boundary conditions using the ghost cells.

(c) Solve, with your code, the following 1-D isothermal hydrodynamics problem. The x-grid goes from $x = -50$ to $x = 50$, the boundary conditions are periodic, the isothermal sound speed is $c_s = 1$. The initial condition is

$$\rho(x, t = 0) = 1 + \exp\left(-\frac{x^2}{200}\right) \tag{15}$$

$$u(x, t = 0) = 0 \tag{16}$$

Plot a few time snapshots: $t = 15$, $t = 30$, $t = 45$ and $t = 60$. Describe your results, and try to explain what you see. NOTE: Use, for simplicity, a fixed time step, but choose it small enough that the algorithm remains stable at all times (i.e. that the CFL condition is met at all times).

(d) *(Voluntary)* Now do the same, but with variable time step. Calculate the Δt_{CFL} at each time step and choose $\Delta t = 0.4 * \Delta t_{\mathrm{CFL}}$ for safety.

(e) *(Voluntary)* Figure out how to produce a movie of your hydrodynamic waves. For this you must write intermediate results to a file after fixed time intervals $\Delta t_{\mathrm{write}}$. Since you have a variable time step you therefore must be clever to assure that the algorithm arrives exactly at those write-times, despite of the a-priori-unknown Δt. Once you have a file containing a sequence of snapshots, produce a sequence of images and use your favorite movie-making facility to make a movie. Please discuss over the moodle (or directly among yourselves) how to make movies (MPEG or AVI or so) from a sequence of still images.