Problem Set 10

Exercises for course Fundamentals of Simulation Methods, WS 2021

Prof. Dr. Mario Flock, Prof. Dr. Friedrich Röpke
Tutors: Brooke Polak (brooke.polak@uni-heidelberg.de | ITA), Glen Hunter (glen.hunter@uni-heidelberg.de | ITA), Jan Henneco (jan.henneco@h-its.org | HITS)
Offices: HITS: Schloss-Wolfsbrunnenweg 35, 69118 Heidelberg; ITA: Albert-Ueberle-Strasse 2, 69120 Heidelberg

Hand in until Wednesday, 26.01.2022, 23:59
Tutorials times: 27-28.01.2022
Group 1: Brooke | Thursday 11:00 - 13:00
Group 2: Glen | Thursday 14:00 - 16:00
Group 3: Jan | Friday 11:00 - 13:00

1. 1D hydrodynamics solver: Riemann/ Sod shock problem

In this exercise you are going to write and use an Eulerian finite-volume code (based on the Godunov method) to solve the 1D equations of hydrodynamics:

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

$$\frac{\partial (\rho u)}{\partial t} + \frac{\partial \left[\rho u^2 + P\right]}{\partial x} = 0\,, \tag{2}$$

$$\frac{\partial (\rho E)}{\partial t} + \frac{\partial [u(\rho E + P)]}{\partial x} = 0 \tag{3}$$

where ρ is the density, u the velocity, and P the pressure. The system is written in conservative form. The set of conservative variables is:

$$\mathbf{q} = \begin{bmatrix} \rho \\ \rho u \\ \rho E \end{bmatrix} \tag{4}$$

and the fluxes are:

$$\mathbf{F}_{Euler} = \begin{bmatrix} \rho u \\ \rho u^2 + P \\ u(\rho E + P) \end{bmatrix} \tag{5}$$

where ρE is the *total energy density*:

$$\rho E \equiv e_{int} + \frac{\rho u^2}{2} \tag{6}$$

and e_{int} is the *internal energy density*.
For our system of equations we need an *adiabatic* equation of state. You can recover the value of the pressure from the total energy density:

$$P = (\gamma - 1)(\rho E - \frac{\rho u^2}{2}) \tag{7}$$

where γ is the adiabatic index. For these simulations we set $\gamma = 1.4$. Here the sound speed, c_s, is defined as:

$$c_s \equiv \sqrt{\frac{\gamma P}{\rho}} \tag{8}$$

1. Construct a finite-volume adiabatic hydrodynamics solver in 1D following the **pseudo-code provided at the end of the sheet**.

   Additional remarks:

   - Use *constant reconstruction* to get the left and right states at each interface $i + 1/2$:

   $$\mathbf{q_L}_{,i+1/2} = \mathbf{q}_i, \quad \mathbf{q_R}_{,i+1/2} = \mathbf{q}_{i+1} \tag{9}$$

   - Use the *HLL* flux to solve the Riemann problem at each interface $i + 1/2$:

   $$\mathbf{F}_{HLL} = \begin{cases} \mathbf{F_L} & if\ S_L \geq 0 \\ \frac{S_R \mathbf{F_L} - S_L \mathbf{F_R} + S_L S_R (\mathbf{q_R} - \mathbf{q_L})}{S_R - S_L} & if\ S_L < 0\ and\ S_R > 0 \\ \mathbf{F_R} & if\ S_R \leq 0 \end{cases}$$

   where $S_L = min(u_L - c_{s_L}, u_R - c_{s_R})$, $S_R = max(u_L + c_{s_L}, u_R + c_{s_R})$[1], $\mathbf{F_L} = \mathbf{F}_{Euler}(\mathbf{q_L})$ and $\mathbf{F_R} = \mathbf{F}_{Euler}(\mathbf{q_R})$.

   - Use the *RK1* algorithm to integrate in time: remember that the time step is limited by the CFL condition:

   $$\Delta t = \text{CFL} \cdot \frac{\Delta x}{(|u| + c_s)_{max}} \tag{10}$$

   - For setting *Dirichlet* boundary conditions, you can simply create 2 extra *ghost cells* and fill the first and second ghost cells with the provided boundary values. In this case there is no need to update the ghost cells at every time step, just remember to fill them before entering the time loop.
   - The conservative variables are stored at cell centers ($N_{cc} = N_x + 2$ ghost cells), the L,R states and HLL fluxes are stored at face centers ($N_{fc} = N_x + 1$).

[1] S_L and S_R are estimates of the wave speeds (given by the eigenvalues of the system) that are needed to solve the Riemann Problem at each interface in order to get the Godunov flux.

- For Python users: if you want to reduce the run time, avoid (the inner) explicit loops and use *numpy.ndarray* operations instead.

Here we are going to consider the Riemann problem, also known as the Sod shock tube. This problem looks at the propigation of a shock wave created by the interface of two fluids of different densities and pressure. Let us look at how the problem is set up. Consider an x-grid that goes from $x = 0$ to $x = 1$ and is divided into $N_x = 100$ cells. At $t = 0$ the system is divided into *left* and *right* states:

- Left state ($x \leq 0.5$): $\rho = 1$, $p = 1$, $u = 0$.
- Right state ($x > 0.5$): $\rho = 0.125$, $p = 0.1$, $u = 0$.

and $\gamma = 1.4$. Dirichlet boundary conditions apply, and the boundary values are given by the initial L/R states[2].

2. Using the initial conditions above, run the finite-volume code to t=0.2 and plot the results for $\rho(x)$, $p(x)$, and $u(x)$ and compare against the semi-analytical solution (found in Riemann.txt where the columns are x, $\rho(x)$, $p(x)$ and $u(x)$).

3. Plot the time evolution of these quantities in a $x - t$ *diagram*, or create an animation of the time evolution. (For instance, you can use *pyplot.imshow*)

4. Redo the problem using $N_x = 1000$.

5. Explain the shape of the solution: where is the contact discontinuity, where is the rarefaction wave, and where is the shock wave?

6. Describe the differences that you observe when increasing the resolution, especially with respect to numerical diffusivity across the rarefaction wave, the shock wave and the contact discontinuity.

7. Experiment with the setup values of the Riemann problem and try to find initial conditions such that the outer (non-linear) sonic waves are both shock waves travelling outwards. Plot the results for $\rho(x)$, $u(x)$, and $p(x)$ at what you think it may be a good final time.

[2]Note that: $\rho E_{L/R}(t = 0) = \dfrac{p_{L/R}(t = 0)}{\gamma - 1}$

Algorithm 1 Finite Volume 1D code

create grid
set initial conditions
t=0
while $t < t_{max}$ **do**
    store $(\rho, \rho u)$ in output
    get Δt using Eq. 10
    apply periodic BCs
    for each interface $i + 1/2$ **do**
        get L,R states at interface using constant reconstruction
        $\mathbf{F}_{i+1/2} = \mathbf{F}_{HLL}(\mathbf{q_L}, \mathbf{q_R})$
    end for
    for each cell i **do**
        get residuals: $\mathbf{R}_i = \frac{\mathbf{F}_{i+1/2} - \mathbf{F}_{i-1/2}}{\Delta x} \Delta t$
        update (RK1): $\mathbf{q}_i = \mathbf{q}_i - \mathbf{R}_i$
    end for
    t = t + Δt
end while