Riemann Problem and Flux Functions

Solving 1D Finite Volume with Ateles

Repetition Finite Volume

- The key idea of the Finite Volume approach is the change of the state by the fluxes across the boundaries of finite volumes
- It therefore resembles a direct discretization of the conservative formulation:

$$u_t + f(u)_x = 0$$

- Here u is the solution of the conservative partial differential equation, and a function of space (x) and time (t)

Continued Finite Volume

- Integration in space yields with $\Delta x = x_1 - x_0$

$$\int_{\Delta x} u_t + f(u)_x \mathrm{dx} = \int_{\Delta x} u_t \mathrm{dx} + \int_{\Delta x} f(u)_x \mathrm{dx}$$

$$= \frac{\partial}{\partial t} \int_{\Delta x} u \mathrm{dx} + f(u(x_1, t)) - f(u(x_0, t)) = 0$$

Integral Mean

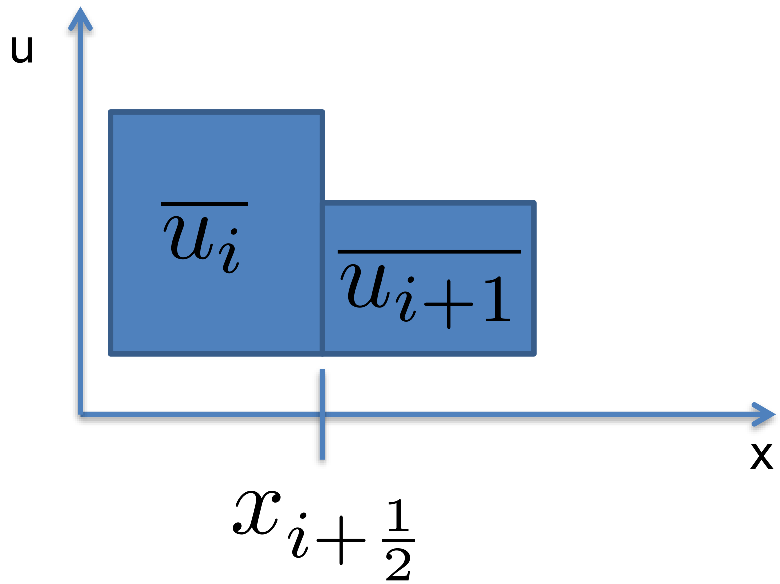
- Introducing the integral mean $\overline{u} = \frac{1}{\Delta x} \int_{\Delta x} u \mathrm{dx}$
- We obtain the semi-discrete form:

$$\frac{\partial \overline{u}}{\partial t} + \frac{1}{\Delta x}(f(u(x_1, t)) - f(u(x_0, t))) = 0$$

- The actual solution u is not available in the scheme, instead, the fluxes have to be approximated in terms of the integral mean

Numerical Flux

- Approximation of the flux on the element edges



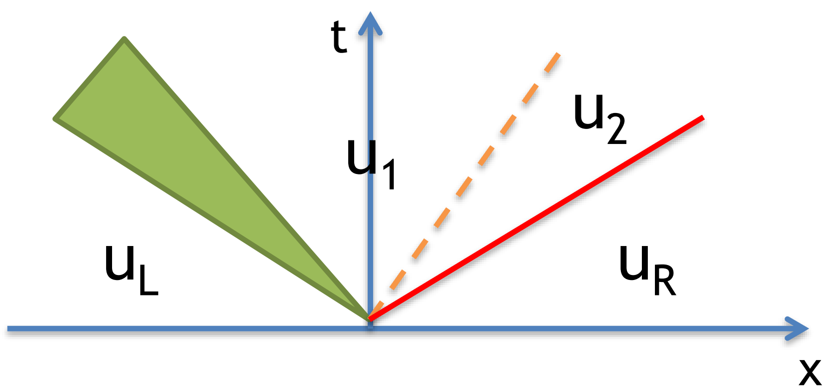
$$g(\overline{u_i}, \overline{u_{i+1}}, t) \approx f(u(x_{i+\frac{1}{2}}, t))$$

Semi-discrete Form with Numerical Fluxes

$$\frac{\partial \overline{u_i}}{\partial t} + \frac{1}{\Delta x}\left(g(\overline{u_i}, \overline{u_{i+1}}, t) - g(\overline{u_{i-1}}, \overline{u_i}, t)\right) = 0$$

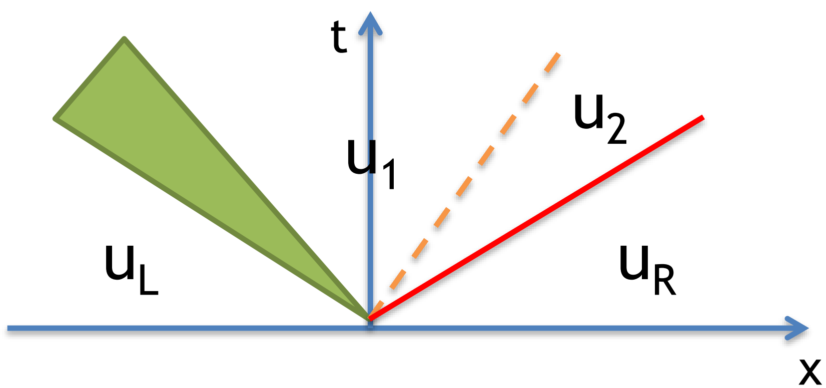
Riemann Problem

- To find the numerical flux, a Riemann problem needs to be solved
- For this, the characteristics are computed and the states split into the corresponding characteristic variables
- The state between the characteristics is found by linear combinations of the characteristic variables



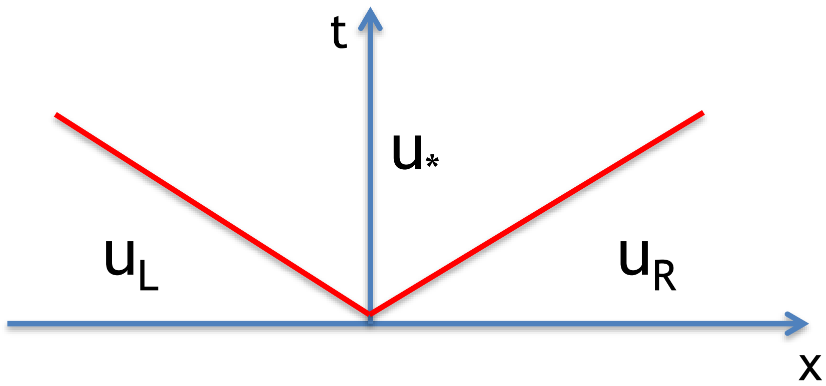
Godunov Flux

- Use the solution of the Riemann problem to find the state on cell edges and use it in the flux computation
- For the nonlinear Euler equations this can only be found iteratively
- Relatively expensive



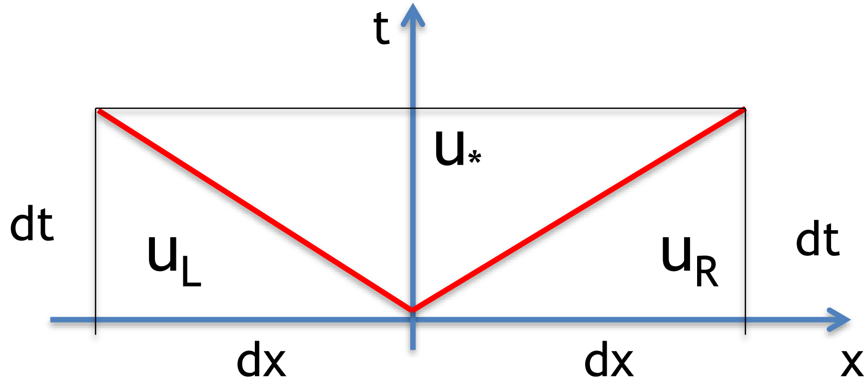
HLL Flux

- Harten, Lax, Leer approximative flux:
  - Linearization
  - Only consider fastest and slowest wave
- Very robust and widely applicable
- Cheaper than iterative computation of the exact problem



Lax-Friedrichs Flux

- Simplest approach just based on the discretization instead of the equation
- Only maximal and minimal wave speed considered
- Overestimated by waves, reaching dx within dt



Fluxes in Ateles

- These are three fluxes, available in Ateles and they can be selected for the Euler equations by:
  - `numflux = 'godunov'`
  - `numflux = 'hll'`
  - `numflux = 'lax_friedrich‘`

- We will have a look at the Sod problem, which is a Riemann problem with a state of density=1, velocity=0 and pressure=1 on the left and a state of density=0.125, velocity=0 and pressure=0.1 on the right.

Split Configuration

- The configuration is split into two parts
  - rp_params.lua contains the definition of the Riemann problem
  - ateles.lua contains further ateles settings

- There is an exact riemann solver, that can produce a reference result, which also makes use of the rp_params.lua settings via the riemann.lua configuration

Extracting data from Ateles

- Individual elements from the simulation can be extracted by the tracking mechanism
- Tracking objects are defined in the tracking table of ateles.lua
- Each one needs a label, folder, variables to track, shape, format and a time_control to state what should be tracked when

- We will use the asciiSpatial format to obtain spatial information for all elements in form of a simple text file

Evaluation

- The produced text files can be visualized with gnuplot

- There is an example script in plot.gnu
- Note, that you might have to adapt the script and especially the file names

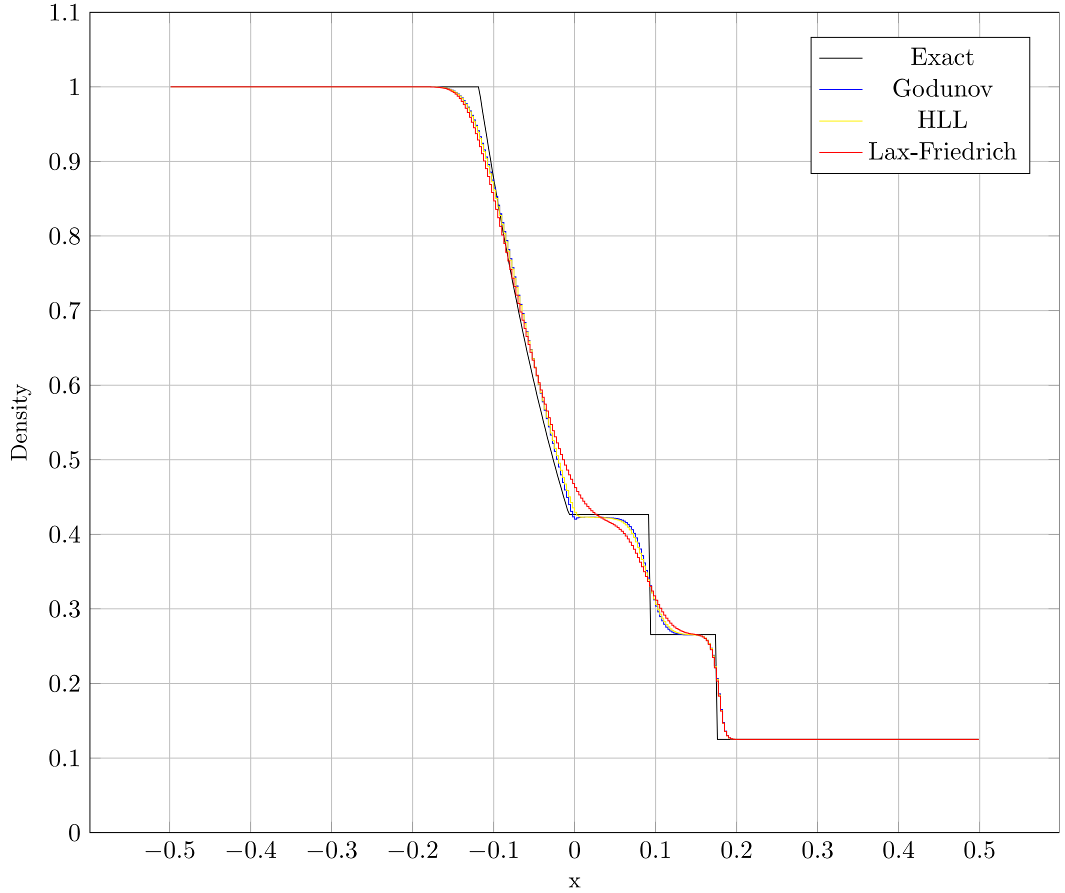
Task

- Copy the data from your local home directory:
  `cp -r $KURS/exercises/hpcfdx3 $MYWS`
- Modify the job script flux.job, as discussed in earlier exercises
- Run the sod problem with different configurations:
  - Vary the numerical flux
  - Vary the number of elements
  - Have a look at the different variables
  - Modify the initial condition to solve a different Riemann problem

Workflow

- Exact solution (see flux.job):
  `$KURS/bin/solve_euler_riemann`
- Run each flux function (see flux.job):
  `export FLUX=<<numflux>>`
  Call gnuplot after each run:
  `gnuplot plot.gnu`
- Choose a meaningful name:
  `mv my-plot.ps <<meaningful-name>>`
- Display plot (on frontend):
  `evince <<meaningful-name>>`

Density Distribution, Different Fluxes, 400 El.



Same Plot for 1000 Elements