Part 2: Introduction to Finite Volume Methods

Mathematisches Institut

August 14, 2017

Outline

1. Idea of upwinding and numerical dissipation
2. CFL condition for stability
3. Godunov Finite Volume method 1D
4. Local Lax-Friedrichs method
5. Finite Volume methods in 2D
6. Comments on advanced techniques

Linear Advection Equation

- We consider simplest hyperbolic PDE: Find $u(x,t) \in \mathbb{C}^1(\mathbb{R})$, such that

$$u_t + \lambda\, u_x = 0, \quad (x,t) \in [a,b] \times [0,T] \tag{1}$$

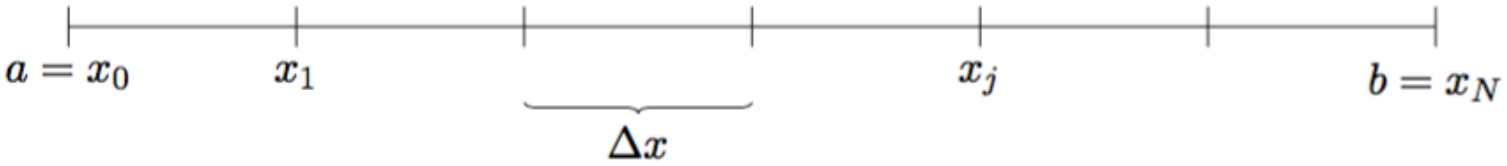
  with constant velocity $\lambda > 0$, periodic boundary conditions and initial condition $u(x, t=0) = u_0(x) \in \mathbb{C}^1(\mathbb{R})$

- From Part 1 we know that the
  - exact solution is given by $u(x,t) = u_0(x - \lambda\, t) \in \mathbb{C}^1(\mathbb{R})$,
  - energy of the solution is constant in time

$$\int_{\mathbb{R}} (u(x,t))^2 dx = \int_{\mathbb{R}} (u_0(x))^2 dx, \forall t \in [0,T]. \tag{2}$$

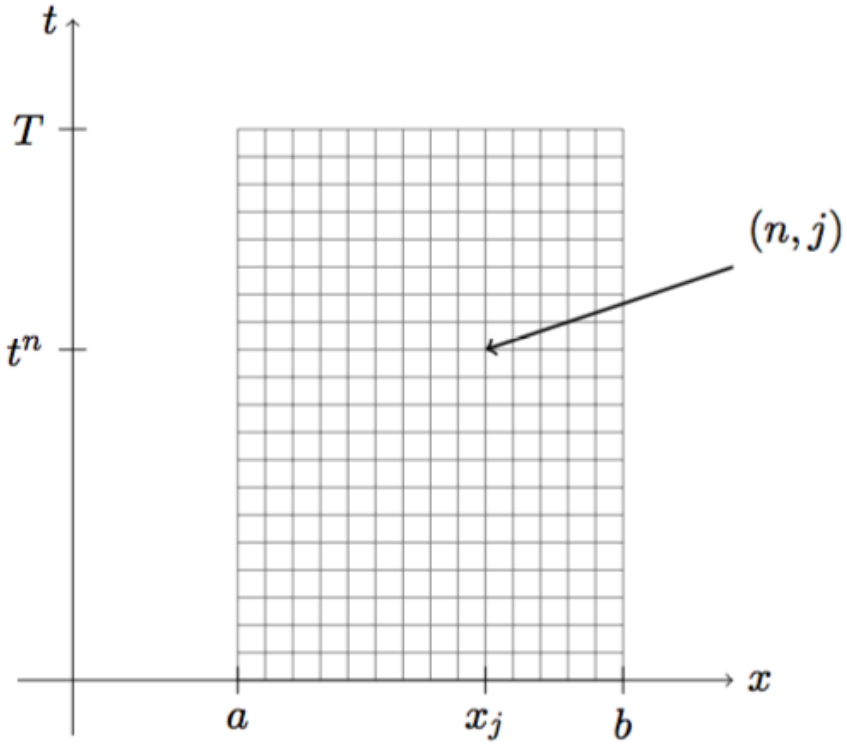
Finite Difference Method

- Discretization of the computational domain
  - Computational domain in space $\Omega = [a, b] \subset \mathbb{R}$
  - Construction of regular grid $x_i = a + i\,\Delta x$ with $\Delta x = (b - a)/N$ for $i = 0, ..., N$ and the number of grid intervals $N \in \mathbb{N}$



Finite Difference Method

- Discretization of the computational domain
  - Computational domain in time $[0, T] \subset \mathbb{R}^+$
  - Construction of regular grid $t^n = n\,\Delta t$ with $\Delta t = T/M$ for $n = 0, ..., M$ and the number of time steps $M \in \mathbb{N}$



Finite Difference Method

- Discrete approximation
  - The unknowns in finite differences are values at the space-time grid nodes (x_i, t^n)
  - Initial condition determines the discrete values at time $t = 0$

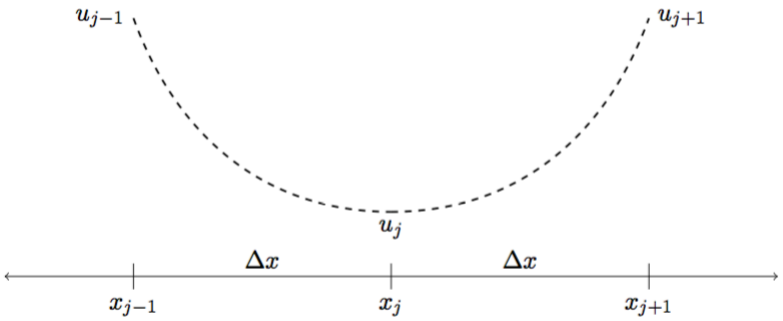
$$U_i^0 = u_0(x_i), \quad \forall i \tag{3}$$

  - Goal is to find a discrete approximation at all other grid nodes

$$U_i^n \approx u(x_i, t^n) \in \mathbb{R}, \quad \forall i, n > 0 \tag{4}$$

Finite Difference Method

- Construction of discrete derivatives u_t, u_x
  - Naive first idea: use central finite differences!
  - Idea is based on Taylor expansion/Polynomial interpolation



  - The spatial derivative at grid node x_i can be approximated by a central finite difference formula

$$u_x\Big|_{x=x_i} \approx \frac{U_{i+1} - U_{i-1}}{2\,\Delta x} \tag{5}$$

  - Analogously, the temporal derivate at time level t^n can be approximated by a one-sided finite difference formula

$$u_t\Big|_{t=t^n} \approx \frac{U^{n+1} - U^n}{\Delta t} \tag{6}$$

Finite Difference Method

- Putting everything together, we get our first numerical method

$$\frac{U_i^{n+1} - U_i^n}{\Delta t} + \lambda \frac{U_{i+1}^n - U_{i-1}^n}{2\,\Delta x} = 0, \tag{7}$$

for $i = 0, ..., N$ and $n = 1, ..., M-1$ with the initial values

$$U_i^0 = U_0(x_i), \quad i = 0, ..., N. \tag{8}$$

and the boundary conditions at the 'ghost indices' $i = -1, N+1$.

Finite Difference Method

- Code structure
  - Choose $\lambda > 0$, N, M
  - Compute Δx, Δt
  - Compute initial values $U_i^0 = u_0(x_i)$
  - Loop over the number of time steps and compute in every step

$$U_i^{n+1} = U_i^n - \frac{\lambda \Delta t}{2\Delta x}(U_{i+1}^n - U_{i-1}^n) \quad i = 0, ..., N, \tag{9}$$

  with the (periodic) boundary conditions

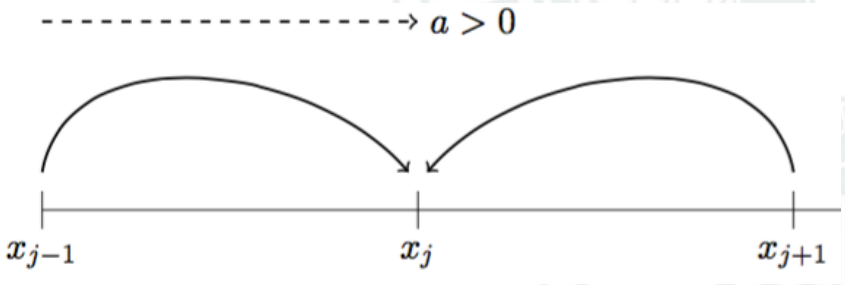
$$U_{-1}^n = U_N^n \text{ and } U_{N+1}^n = U_0^n \tag{10}$$

  - Note, only the current solution array needs to be stored!
- Numerical example (demonstration)

$$u_0(x) = \sin(2\,\pi\,x), \quad x \in \Omega = [0, 1] \tag{11}$$

Finite Difference Method

- Why did it fail? A physical explanation....
  - The physics of the problem is that information is transported along the characteristics from left to right ($\lambda > 0$)



$\Rightarrow$ **The central finite difference approximation takes information from the wrong direction, against the physics of the problem!**

Finite Difference Method

- Why did it fail? A mathematical explanation....
  - We know that for linear advection, the energy $E(t) = \int_{\mathbb{R}} (u(x,t))^2 dx$ is constant in time
  - We can compute the discrete energy of the approximate solution $E^n = \sum_{i=0}^{N} (U_i^n)^2 \Delta x$
  - We can proof the following <u>Theorem</u>: For the approximate solution computed with the central finite difference scheme and the explicit Euler time integration, Eq. (7), the discrete energy evolves as

$$E^{n+1} = E^n + \frac{\Delta x}{2} \sum_{i=0}^{N} (U_i^{n+1} - U_i^n)^2 \tag{12}$$

Finite Difference Method

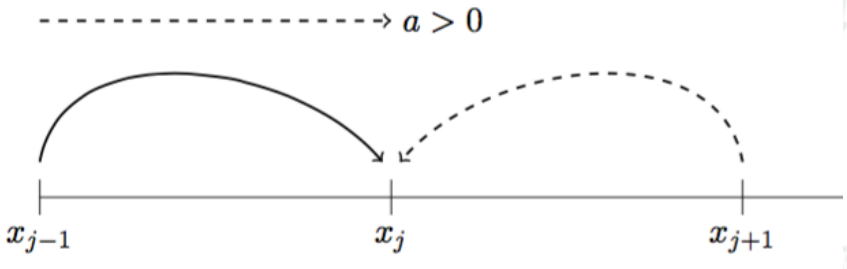
- Why did it fail? A mathematical explanation....
  - We know that for linear advection, the energy $E(t) = \int_{\mathbb{R}} (u(x,t))^2 dx$ is constant in time
  - We can compute the discrete energy of the approximate solution $E^n = \sum_{i=0}^{N} (U_i^n)^2 \Delta x$
  - We can proof the following <u>Theorem</u>: For the approximate solution computed with the central finite difference scheme and the explicit Euler time integration, Eq. (7), the discrete energy evolves as

$$E^{n+1} = E^n + \frac{\Delta x}{2} \sum_{i=0}^{N} (U_i^{n+1} - U_i^n)^2 \tag{12}$$

$\Rightarrow$ **The discrete energy E^n grows in every time step, independent of the choice of Δx, Δt.**

Finite Difference Method

- Construction of discrete derivatives u_t, u_x according to the physics of the problem



- The simplest possibility for $\lambda > 0$ reads as

$$\frac{U_i^{n+1} - U_i^n}{\Delta t} + \lambda \frac{U_i^n - U_{i-1}^n}{\Delta x} = 0, \tag{13}$$

and is called **Upwind Finite Difference Method**

Finite Difference Method

- Physical interpretation of the Upwind FDM
  - Reformulation gives

$$\frac{U_i^{n+1} - U_i^n}{\Delta t} + \lambda \frac{U_i^n - U_{i-1}^n}{\Delta x} = 0,$$
$$\Rightarrow \frac{U_i^{n+1} - U_i^n}{\Delta t} + \lambda \frac{U_{i+1}^n - U_{i-1}^n}{2\,\Delta x} = \frac{\lambda\,\Delta x}{2} \frac{U_{i+1}^n - 2\,U_i^n + U_{i-1}^n}{\Delta x^2},$$

  - The additional term is...

$$\frac{\lambda\,\Delta x}{2} \frac{U_{i+1}^n - 2\,U_i^n + U_{i-1}^n}{\Delta x^2} \approx \frac{\lambda\,\Delta x}{2} u_{xx} \tag{14}$$

...an approximation to diffusion!

$\Rightarrow$ **This is artificial diffusion, introduced by the upwinding**

Finite Difference Method

- Physical interpretation of the Upwind FDM
  - Approximately, the Upwind FDM solves the (artificial) advection-diffusion equation

$$u_t + a\, u_x = \frac{\lambda\, \Delta x}{2}\, u_{xx} \tag{15}$$

  - The physical effect of diffusion is to 'damp' and 'smooth'

    $\Rightarrow$ **Artificial (or numerical) diffusion (or dissipation) plays a crucial role for the stability of numerical methods applied to hyperbolic partial differential equations.**

  - Numerical example (demonstration)

$$u_0(x) = \sin(2\,\pi\, x) \in \Omega = [0, 1] \tag{16}$$

Finite Difference Method

- When does it work?
  - We know that for linear advection, the energy $E(t) = \int_{\mathbb{R}} (u(x,t))^2 dx$ is constant in time
  - We can proof the following Theorem: For the approximate solution computed with the Upwind Finite Difference scheme and the explicit Euler time integration, eq. (13), the discrete energy evolves as

$$E^{n+1} = E^n + \left(\frac{\lambda^2 \Delta t^2}{\Delta x} - \lambda \Delta t \right) \sum_{i=0}^{N} (U_i^n - U_{i-1}^n)^2. \qquad (17)$$

Finite Difference Method

- When does it work?
  - We know that for linear advection, the energy $E(t) = \int_{\mathbb{R}} (u(x,t))^2 dx$ is constant in time
  - We can proof the following Theorem: For the approximate solution computed with the Upwind Finite Difference scheme and the explicit Euler time integration, eq. (13), the discrete energy evolves as

$$E^{n+1} = E^n + \left(\frac{\lambda^2 \Delta t^2}{\Delta x} - \lambda \Delta t \right) \sum_{i=0}^{N} (U_i^n - U_{i-1}^n)^2. \qquad (17)$$

$\Rightarrow$ **The discrete energy E^n decays in every time step, if $\left(\frac{\lambda^2 \Delta t^2}{\Delta x} - \lambda \Delta t \right) \leq 0$.**

Finite Difference Method

- When does it work?
  - **The CFL (Courant, Friedrichs, Lewy) condition**

$$\frac{\lambda^2 \Delta t^2}{\Delta x} \leq \lambda \Delta t$$
$$\frac{\lambda \Delta t}{\Delta x} \leq 1 \tag{18}$$

or introducing the CFL number $\boldsymbol{CFL} \leq \mathbf{1}$, we get

$$\boldsymbol{\Delta t = CFL \frac{\Delta x}{\lambda}} \tag{19}$$

Finite Difference Method

- Important intermediate conclusion
  - Numerical dissipation is **good**, as it enables stability
  - Numerical dissipation is **bad**, as it introduced numerical errors (damping) of the solution

  $\Rightarrow$ **Optimal numerical dissipation is as small as possible, but sufficient to give stability!**

Finite Difference Method

- Extension to non-linear hyperbolic PDEs

$$u_t + f_x = 0, \tag{20}$$

with the flux function $f = f(u)$.

  - Quasilinear form

$$u_t + \lambda(u)\, u_x = 0, \tag{21}$$

    with the non-linear advection speed $\lambda(u) = \frac{df}{du}$.

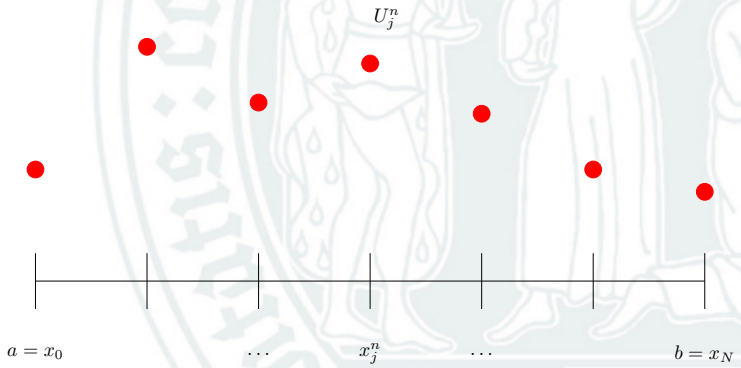
  - Advection speed depends on the solution
    $\Rightarrow$ **Upwinding direction depends on the solution!**

  - From Part I: non-linear hyperbolic PDEs generate shocks/discontinuities
    $\Rightarrow$ **Non-smooth solutions, no Taylor expansion!**

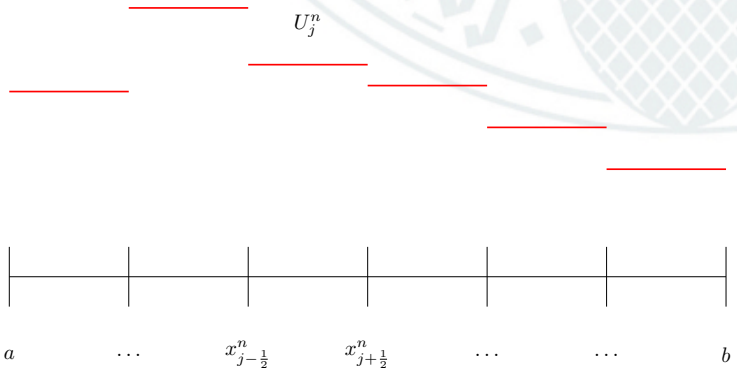
$\Rightarrow$ **Extension of classic FDM to non-linear problems is not straight forward!**

Godunov

- **Godunov's first idea**
  - Re-interpretation of the grid and of the approximation: No grid nodes, no grid node values $U_i^n \approx u(x_i, t^n)$



  - Instead, **grid cells** and **mean values** $U_i^n \approx \frac{1}{\Delta x} \int_{x_{i-1/2}}^{x_{i+1/2}} u(x, t^n)\, dx$



Integral form of the PDE

- We need an evolution equation for the mean values!
- Start from the PDE and integrate over the grid cell and the time step

$$u_t + f_x = 0$$

$$\int_{t^n}^{t^{n+1}} \int_{x_{i-1/2}}^{x_{i+1/2}} u_t + f_x \, dxdt = 0 \tag{22}$$

Integral form of the PDE

- Integration by parts and definition of mean values

$$\int_{t^n}^{t^{n+1}} \int_{x_{i-1/2}}^{x_{i+1/2}} u_t \, dx \, dt = \int_{x_{i-1/2}}^{x_{i+1/2}} u(x, t^{n+1}) - u(x, t^n) \, dx \tag{23}$$
$$= \Delta x (U_i^{n+1} - U_i^n)$$

- Same for the flux part

$$\int_{t^n}^{t^{n+1}} \int_{x_{i-1/2}}^{x_{i+1/2}} f_x dx \, dt = \int_{t^n}^{t^{n+1}} f(x_{i+1/2}, t) - f(x_{i-1/2}, t) \, dt \tag{24}$$
$$= \Delta t (\bar{F}_{i+1/2}^n - \bar{F}_{i-1/2}^n)$$

Integral form of the PDE

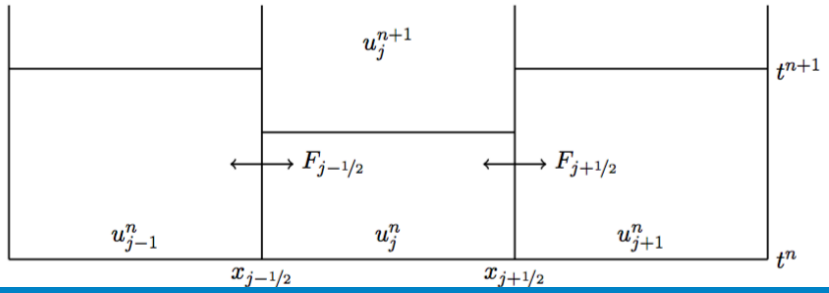
- **Exact evolution equation of the mean values**

$$U_i^{n+1} = U_i^n - \frac{\Delta t}{\Delta x}(\bar{F}_{i+1/2}^n - \bar{F}_{i-1/2}^n) \tag{25}$$

- **Problem**
  - Definition of the integral fluxes

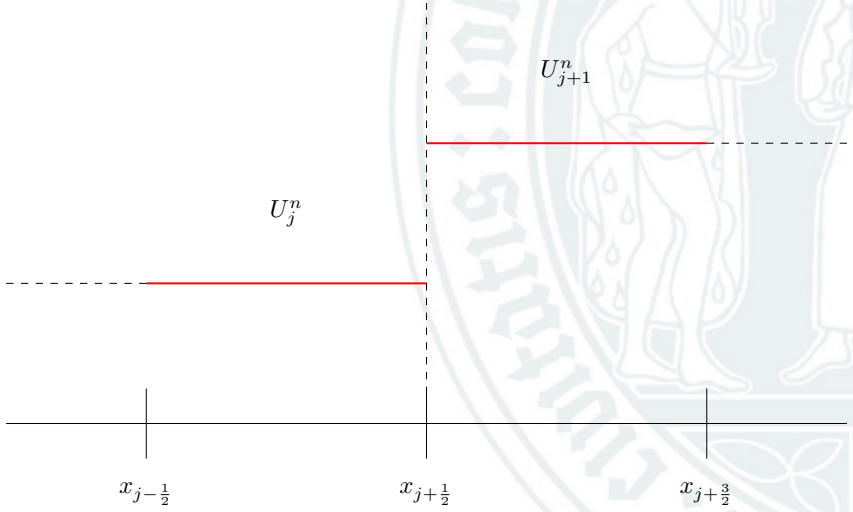
$$\bar{F}_{i+1/2}^n := \frac{1}{\Delta t}\int_{t^n}^{t^{n+1}} f(x_{i+1/2}, t)dt = \frac{1}{\Delta t}\int_{t^n}^{t^{n+1}} f(u(x_{i+1/2}, t))dt \tag{26}$$

    depends on the solution u!
  - We only have mean values U_i^n, which 'jump' at $x_{i+1/2}$



Godunov

- **Godunov's second idea**
  - Interpretation of the interface jump as a **Riemann problem**



$$u_t + f_x = 0, \quad u_0(x) = \begin{cases} u_i^n, & \text{for } x < x_{i+1/2} \\ u_{i+1}^n, & \text{for } x > x_{i+1/2} \end{cases} \tag{27}$$

$\Rightarrow$ **Riemann solution: physics of the information at the interfaces (generalised non-linear upwinding!)**

Godunov

- **Godunov's second idea**
  - Interpretation of the interface jump as a **Riemann problem**
  - Compute the exact solution of the Riemann problem $u_{RP}(x,t)$ at each interface
  - From Part I we know that Riemann problem solutions consists of shocks and rarefaction waves and are constant along characteristics $x'(t) = const$
  - Use the Riemann solution to approximate the integral flux at the interface and compute a **numerical flux**

$$\bar{F}^n_{i+1/2} \approx F^{n,GOD}_{i+1/2} = \frac{1}{\Delta t} \int_{t^n}^{t^{n+1}} f(u_{RP}(x_{i+1/2},t))dt = f(u_{RP}(x_{i+1/2},0^+))$$

Godunov Finite Volume Scheme

- Code structure:
  - Choose number of grid cells N
  - Compute Δx and initial mean values U_i^0 from the initial condition
  - Loop over the number of time steps and compute in every step
    - Use CFL-type condition to compute Δt
    - Use (periodic) boundary condition for cells $i = -1, N+1$
    - Compute the Riemann solutions u_{RP} at each interface $x_{i+1/2}$
    - Compute the numerical fluxes $F_{i+1/2}^{n,GOD}$ at each interface
    - Compute the mean values at the new time step

$$U_i^{n+1} = U_i^n - \frac{\Delta t}{\Delta x}(F_{i+1/2}^{n,GOD} - F_{i-1/2}^{n,GOD}) \tag{28}$$

Godunov Finite Volume Scheme

- Properties of the scheme:
  - The Godunov scheme is stable, under the CFL-type condition

$$\Delta t = CFL \frac{\Delta x}{\lambda_{max}} \tag{29}$$

    with the *CFL* number smaller or equal to 1.
  - The maximum signal speed depends on the problem:
    - For linear advection, $\lambda_{max} = \lambda$
    - For non-linear scalar problems, $\lambda_{max} = \max(|\frac{df}{du}|)$
    - For systems, we have to compute the maximum magnitudes of the Eigenvalues of the flux Jacobian matrix
  - For linear advection, the Godunov scheme is similar to the upwind scheme $\Rightarrow$ **natural extension to non-linear problems!**

Godunov -Type Finite Volume Methods

- The exact solution of the Riemann problem at every interface is very expensive!
- Breakthrough of Finite Volume Methods:
  - It is not necessary to compute the Riemann problem exactly
  - It is possible to compute the Riemann problem approximatively
  - **There are many ideas and different ways of computing an approximative Riemann solver: Roe, Lax-Friedrichs, HLL, Flux-Vector Splitting, ...**
- The simplest example: **The local Lax-Friedrichs numerical flux**

$$\bar{F}^n_{i+1/2} \approx F^{n,LLF}_{i+1/2} = \frac{1}{2}\left(f(U^n_i) + f(U^n_{i+1})\right) - \frac{\lambda_{i+1/2,max}}{2}(U^n_{i+1} - U^n_i)$$

  with the maximum interface signal speed $\lambda_{i+1/2,max}$
- This flux is also known as Rusanov flux

Local Lax-Friedrichs Finite Volume Scheme

- Code structure:
  - Choose number of grid cells N
  - Compute Δx and initial mean values U_i^0 from the initial condition
  - Loop over the number of time steps and compute in every step
    - Use CFL-type condition to compute Δt
    - Use (periodic) boundary condition for cells $i = -1, N+1$
    - Compute the numerical fluxes $F_{i+1/2}^{n,LLF}$ at each interface
    - Compute the mean values at the new time step

$$U_i^{n+1} = U_i^n - \frac{\Delta t}{\Delta x}(F_{i+1/2}^{n,LLF} - F_{i-1/2}^{n,LLF}) \tag{30}$$

- Properties
  - **Godunov-type schemes typically have larger numerical dissipation compared to the original Godunov method, but are faster per grid cell and simpler to code!**

$\Rightarrow$ Our projects are based on the LLF-FV scheme!

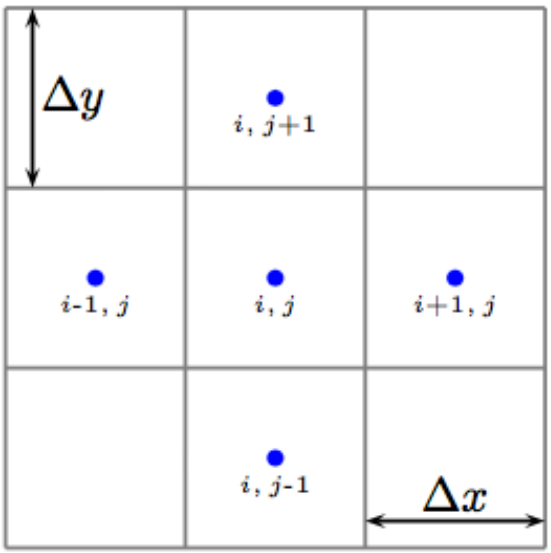
Finite Volume Methods in 2D

- We consider the 2D hyperbolic PDE

$$u_t + \vec{\nabla} \cdot \vec{f} = 0, \quad \boldsymbol{u_t + f_x + g_y = 0} \tag{31}$$

  with the flux functions $f = f(u)$ and $g = g(u)$, initial condition and periodic boundary conditions

- We make the restriction that we consider Cartesian grids



- This allows us to directly extend the 1D ideas to 2D

Integral form of PDE in 2D

- We define the mean value in 2D as

$$U_{i,j}^n = \frac{1}{\Delta x \, \Delta y} \int_{x_{i-1/2}}^{x_{i+1/2}} \int_{y_{j-1/2}}^{y_{j+1/2}} u(x, y, t^n) \, dx \, dy \tag{32}$$

- Integration of the PDE over the grid cell and in time

$$\int_{t^n}^{t^{n+1}} \int_{x_{i-1/2}}^{x_{i+1/2}} \int_{y_{j-1/2}}^{y_{j+1/2}} u_t + f_x + g_y \; dx \, dy \, dt = 0 \tag{33}$$

Integral form of PDE in 2D

- Integration by parts and definition of the mean values

$$\int_{t^n}^{t^{n+1}} \int_{x_{i-1/2}}^{x_{i+1/2}} \int_{y_{j-1/2}}^{y_{j+1/2}} u_t \, dx \, dy \, dt = \Delta x \, \Delta y \, (U_{i,j}^{n+1} - U_{i,j}^{n}) \qquad (34)$$

- Again for the flux part in x-direction

$$\int_{t^n}^{t^{n+1}} \int_{x_{i-1/2}}^{x_{i+1/2}} \int_{y_{j-1/2}}^{y_{j+1/2}} f_x dx \, dy \, dt = \Delta y \, \Delta t \, (\bar{F}_{i+1/2,j}^{n} - \bar{F}_{i-1/2,j}^{n}) \quad (35)$$

and y-direction

$$\int_{t^n}^{t^{n+1}} \int_{x_{i-1/2}}^{x_{i+1/2}} \int_{y_{j-1/2}}^{y_{j+1/2}} g_y dx \, dy \, dt = \Delta x \, \Delta t \, (\bar{G}_{i,j+1/2}^{n} - \bar{G}_{i,j-1/2}^{n}) \quad (36)$$

Integral form of PDE in 2D

- **Exact evolution equation of mean values in 2D**

$$U_{i,j}^{n+1} = U_{i,j}^{n} - \frac{\Delta t}{\Delta x}(\bar{F}_{i+1/2,j}^{n} - \bar{F}_{i-1/2,j}^{n}) - \frac{\Delta t}{\Delta y}(\bar{G}_{i,j+1/2}^{n} - \bar{G}_{i,j-1/2}^{n}) \tag{37}$$

with the mean value of the interface flux

$$\bar{F}_{i+1/2,j}^{n} = \frac{1}{\Delta t\,\Delta y} \int_{t^n}^{t^{n+1}} \int_{y_{j-1/2}}^{y_{j+1/2}} f(u(x_{i+1/2}, y, t))\, dy\, dt \tag{38}$$

- Again, we need to approximate the fluxes with numerical flux functions, e.g. Godunov, LLF, Roe, HLL,...

Finite Volume Methods in 2D

- Local Lax-Friedrichs numerical flux

$$F^{n,LLF}_{i+1/2,j} = \frac{1}{2}(f(u^n_{i,j}) + f(u^n_{i+1,j})) - \frac{\lambda^x_{i+1/2,j,max}}{2}(U^n_{i+1,j} - U^n_{i,j})$$

$$G^{n,LLF}_{i,j+1/2} = \frac{1}{2}(g(u^n_{i,j}) + g(u^n_{i,j+1})) - \frac{\lambda^y_{i,j+1/2,max}}{2}(U^n_{i,j+1} - U^n_{i,j})$$

- With the signal speeds λ^x, λ^y in x-direction and y-direction as the maximum Eigenvalues of the flux Jacobians df/du and dg/du respectively
- **LLF-FV scheme**

$$U^{n+1}_{i,j} = U^n_{i,j} - \frac{\Delta t}{\Delta x}(F^{n,LLF}_{i+1/2,j} - F^{n,LLF}_{i-1/2,j}) - \frac{\Delta t}{\Delta y}(G^{n,LLF}_{i,j+1/2} - G^{n,LLF}_{i,j-1/2})$$

Local Lax-Friedrichs Finite Volume Scheme in 2D

- Code structure:
  - Choose number of grid cells N
  - Compute $\Delta x, \Delta y$ and initial mean values $U^0_{i,j}$ from the initial condition
  - Loop over the number of time steps and compute in every step
    - Use CFL-type condition to compute Δt
    - Use (periodic) boundary condition for all grid cells with indices $i, j = -1$ and/or $i, j = N + 1$
    - Compute the numerical fluxes $F^{n,LLF}_{i+1/2,j}$ and $G^{n,LLF}_{i,j+1/2}$ at each interface
    - Compute the mean values at the new time step

$$U^{n+1}_{i,j} = U^n_{i,j} - \frac{\Delta t}{\Delta x}(F^{n,LLF}_{i+1/2,j} - F^{n,LLF}_{i-1/2,j}) - \frac{\Delta t}{\Delta y}(G^{n,LLF}_{i,j+1/2} - G^{n,LLF}_{i,j-1/2})$$

- CFL condition in 2D with Cartesian regular grids

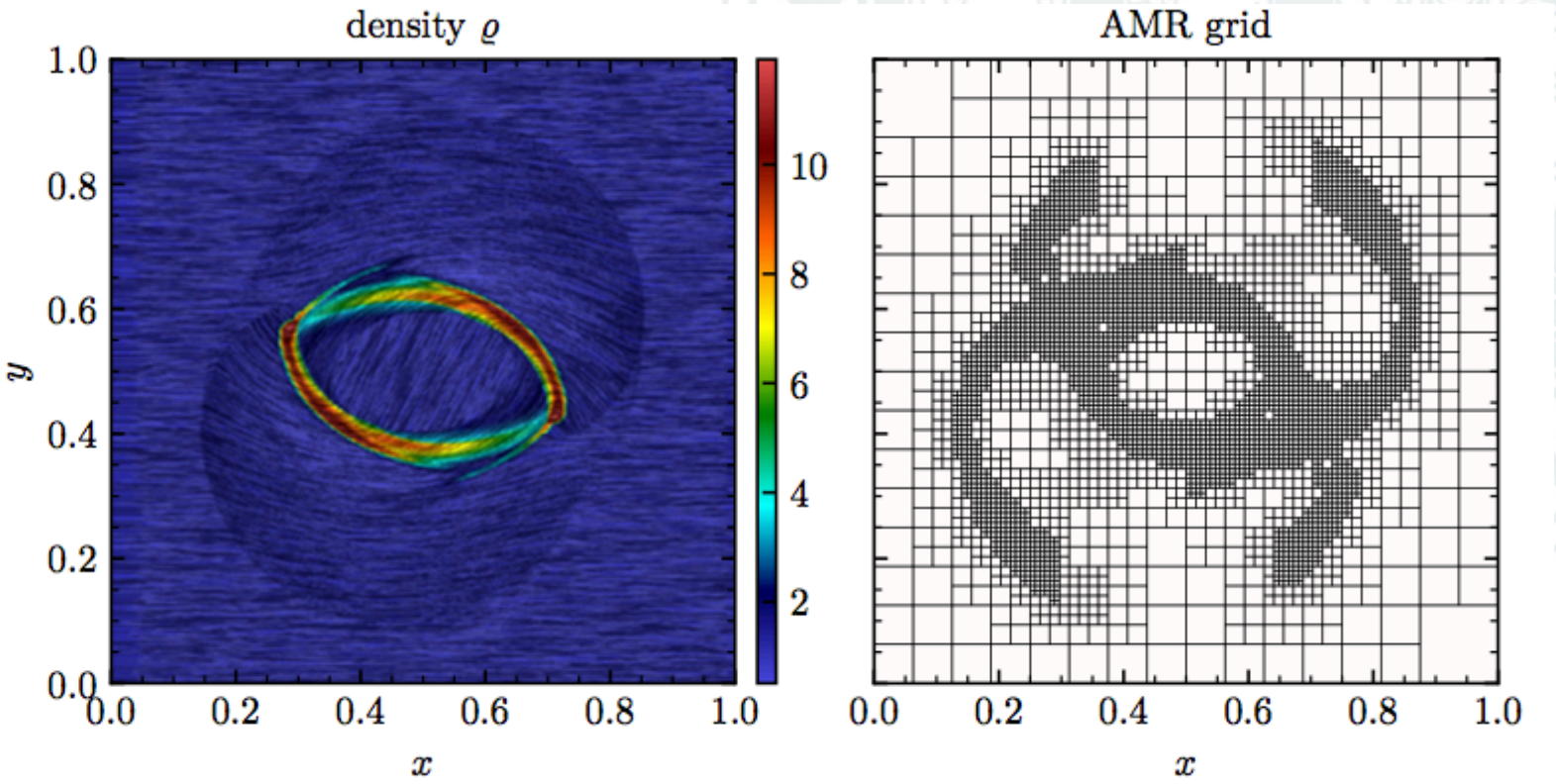
$$\Delta t = CFL \min\left(\frac{\Delta x}{\lambda^x_{max}}, \frac{\Delta y}{\lambda^y_{max}}\right) \tag{39}$$

Comments on the Content of the Project

- Finite Volume methods are the basis of many state-of-the-art solvers for hydrodynamics, gas dynamics, magnetohydrodynamics, ...
- The algorithms drastically speed up when a higher programming language (e.g. C, FORTRAN) is used
- **For real production codes used in natural sciences and engineering, some additional advanced techniques are typically used to drastically increase the efficiency of the FV method!**
- The advanced techniques are out of scope of this 2-day workshop, but it is important to note that these are essential for a competitive simulation framework!

Advanced techniques: AMR

- Adaptive mesh refinement
- We use smaller grid cells at sharp shock fronts, or other small scale features (e.g. turbulence)

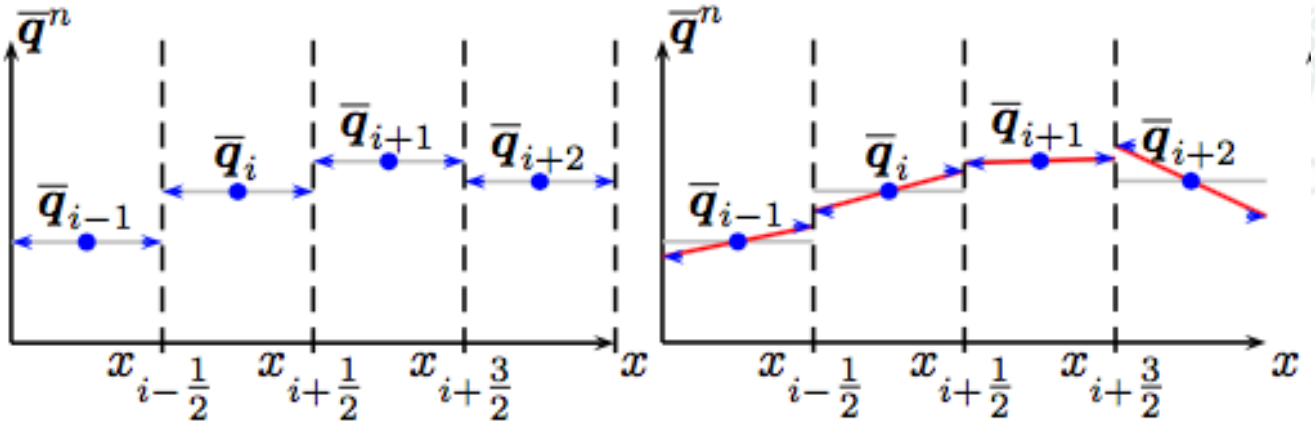


Advanced techniques: HPC

- High performance computing
- Real life problems are time dependent and 3D
- Shocks, turbulence...many scales in space and time
- Extreme high resolution is necessary: millions and billions of grid cells
- FV methods are very well suited for parallel computing on modern super computing clusters
- Scaling on up to $> O(10^5)$ processors is possible (and necessary)!

Advanced techniques: HOM

- High order methods
- **In the projects, we will see that the convergence rate of the FV method is first order only!**
- The numerical diffusion for turbulence can be very high (even more grid cells necessary!)
- It is possible to introduce a high order reconstruction, e.g. piecewise linear to reduce the dissipation



Advanced techniques

- Putting everything together gives a state of the art solver
- If interested in more details and background information, we refer to the DMV article available for download
- **With such a code, simulations with extreme fidelity can be computed (e.g. the Orzag-Tang vortex from the bonus project)**