Lecture 06 - Finite volumes

Derivation, MUSCL, Riemann Solvers

Georgii Oblapenko

1 Objectives

In this lecture, we will learn (or revisit) "classical" finite volume methods for hyperbolic PDEs. We will focus on the following parts of the theory:

1. Motivation, general formulation
2. Exact and approximate Riemann solvers
3. Higher-order reconstruction, slope limiters

2 Motivation, general formulation

We want to solve hyperbolic PDEs of the form

$$\partial_t \mathbf{u} + \nabla \cdot \mathbf{f}(\mathbf{u}) = \mathbf{s}(\mathbf{u}),$$

where $\mathbf{u}$ is the vector of conserved variables, $\mathbf{f}$ is the **flux function**, and $\mathbf{s}$ is the source term.

2.1 Examples: linear advection

Simplest hyperbolic PDE:

$$\partial_t u + \mathbf{a} \cdot \nabla u = 0,$$

i.e. $\mathbf{u} = u$, $\mathbf{f}(\mathbf{u}) = \mathbf{a}u$, $\mathbf{s}(\mathbf{u}) = 0$.

2.2 Examples: inviscid Burgers' equation

Non-linear equation that can exhibit formation of shocks:

$$\partial_t u + \partial_x \left(\frac{u^2}{2} \right) = 0,$$

i.e. $\mathbf{u} = u$, $\mathbf{f}(\mathbf{u}) = \frac{u^2}{2}$, $\mathbf{s}(\mathbf{u}) = 0$.

2.3 Examples: Compressible Euler equations in 2D

$$\partial_t \rho + \nabla \cdot (\rho \mathbf{v}) = 0, \partial_t (\rho \mathbf{v}) + \nabla \cdot (\rho \mathbf{v} \otimes \mathbf{v}) + \nabla p = 0, \partial_t E + \nabla \cdot ((E+p)\mathbf{v}) = 0,$$

i.e. $\mathbf{u} = (\rho, \rho\mathbf{v}, E)^T$, $\mathbf{f}(\mathbf{u}) = (f_x, f_y)$, $\mathbf{s}(\mathbf{u}) = 0$, flux components f_x, f_y are given by

$$f_x = (\rho v_x, \rho v_x^2 + p, \rho v_x v_y, (E+p)v_x)^T, f_y = (\rho v_y, \rho v_x v_y, \rho v_y^2 + p, (E+p)v_y)^T$$

Closure:

$$E = \rho \varepsilon_{int} + \rho \frac{v^2}{2}, \quad p = p(\rho, \varepsilon_{int})$$

2.4 Why methods for unsteady equations?

2.5 Finite volumes

Compared to finite differences, FVM allows for

- complex geometries and meshes
- easier handling of discontinuities
- more robust conservation properties

3 Finite volume method

$$\partial_t \mathbf{u} + \nabla \cdot \mathbf{f}(\mathbf{u}) = \mathbf{s}(\mathbf{u})$$

defined on domain V. Decompose V into control volumes:

$$V = \bigcup_{i=1}^{N} V_i, \;\; V_i \cap V_j = \emptyset, \; i \neq j.$$

3.1 Integral conservation law

Integrate PDE over control volume V_i:

$$\int_{V_i} \partial_t \mathbf{u} \, \mathrm{d}V + \int_{V_i} \nabla \cdot \mathbf{f}(\mathbf{u}) = \int_{V_i} \mathbf{s}(\mathbf{u}) \, \mathrm{d}V.$$

Transform the flux term using the divergence theorem:

$$\int_{V_i} \partial_t \mathbf{u} \, \mathrm{d}V + \int_{\partial V_i} \mathbf{f}(\mathbf{u}) \cdot \mathbf{n} \, \mathrm{d}S = \int_{V_i} \mathbf{s}(\mathbf{u}) \, \mathrm{d}V,$$

or

$$\partial_t \int_{V_i} \mathbf{u} \, \mathrm{d}V + \int_{\partial V_i} \mathbf{f}(\mathbf{u}) \cdot \mathbf{n} \, \mathrm{d}S = \int_{V_i} \mathbf{s}(\mathbf{u}) \, \mathrm{d}V,$$

3.2 Fluxes at cell interfaces

Split the cell boundary ∂V_i into cell faces: $\partial V_i = \cup_j f_{ij}$, $j \in \mathcal{N}(i)$, $\mathcal{N}(i)$ is the set of neighbors of cell i, $f_{ij} = \partial V_i \cap \partial V_j$.

Then we have

$$\int_{\partial V_i} \mathbf{f}(\mathbf{u}) \cdot \mathbf{n} = \sum_{j \in \mathcal{N}(i)} \int_{f_{ij}} \mathbf{f}(\mathbf{u}) \cdot \mathbf{n} \, \mathrm{d}S.$$

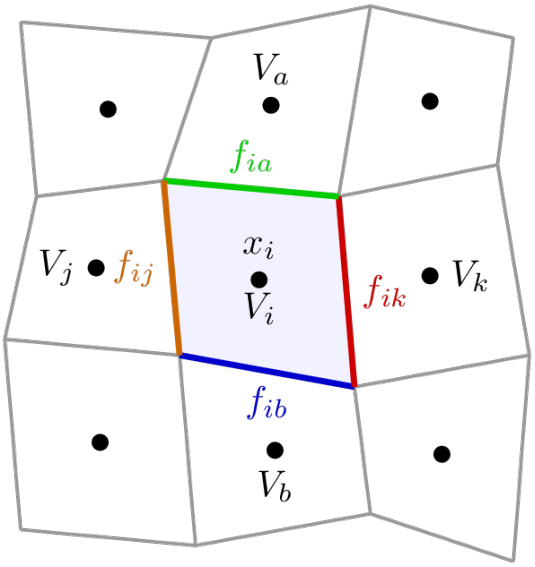

Figure 1: Cells and cell faces

3.3 Fluxes at cell interfaces

We can approximate the flux integral over a cell face f_{ij} as

$$\int_{f_{ij}} \mathbf{f}(\mathbf{u}) \cdot \mathbf{n} \, \mathrm{d}S \approx g_{ij}(\mathbf{u}_i, \mathbf{u}_j),$$

g_{ij} is a numerical approximation of the flux **from cell i into cell j**.

Approximations made?

We made two approximations here, which ones?

Approximations made

We made two approximations here: 1) we approximate integration over the interface f_{ij} with a single-point quadrature 2) we are also (potentially) approximating the function (flux) we are integrating!

Constraints on numerical flux

g_{ij} should satisfy two requirements:

1. **Flux conservation**: $g_{ij}(\mathbf{u}_i, \mathbf{u}_j) = -g_{ji}(\mathbf{u}_j, \mathbf{u}_i)$.
2. **Consistency**: $g_{ij}(\mathbf{u}, \mathbf{u}) = \int_{f_{ij}} \mathbf{f}(\mathbf{u}) \cdot \mathbf{n} \, \mathrm{d}S$.

3.4 Final semi-discrete form

Define

$$\mathbf{u}_i = \frac{1}{V_i} \int_{V_i} \mathbf{u} \, \mathrm{d}V,$$

then we have the **semi-discrete** form of the conservation law:

$$\partial_t \mathbf{u}_i + \frac{1}{V_i} \sum_{j \in \mathcal{N}(i)} g_{ij}(\mathbf{u}_i, \mathbf{u}_j) = \mathbf{s}_i.$$

Source term treatment

Simplest way to treat source term is to simply evaluate it from $\mathbf{u}_i$ and incorporate it independently of the fluxes (fluxes are not directly affected by source terms).

3.5 Cell-centered vs node/vertex-centered

Assume we have a grid we would like to compute our solution on. What does that mean exactly?

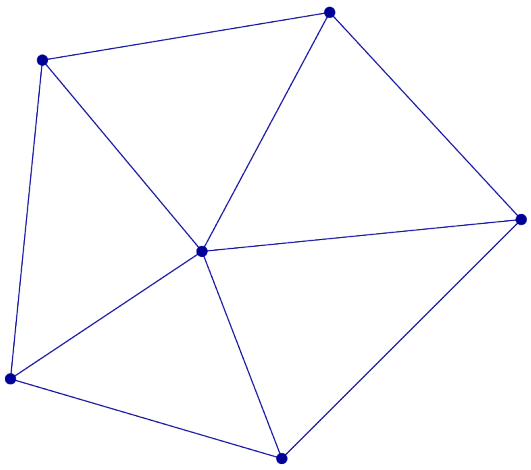

Figure 2: Cells and cell faces

There are two ways to take a grid and define a solution on it: **cell-centered** and **vertex-centered**.

3.6 Cell-centered

In a cell-centered approach, we define the solution at the center of the grid elements.

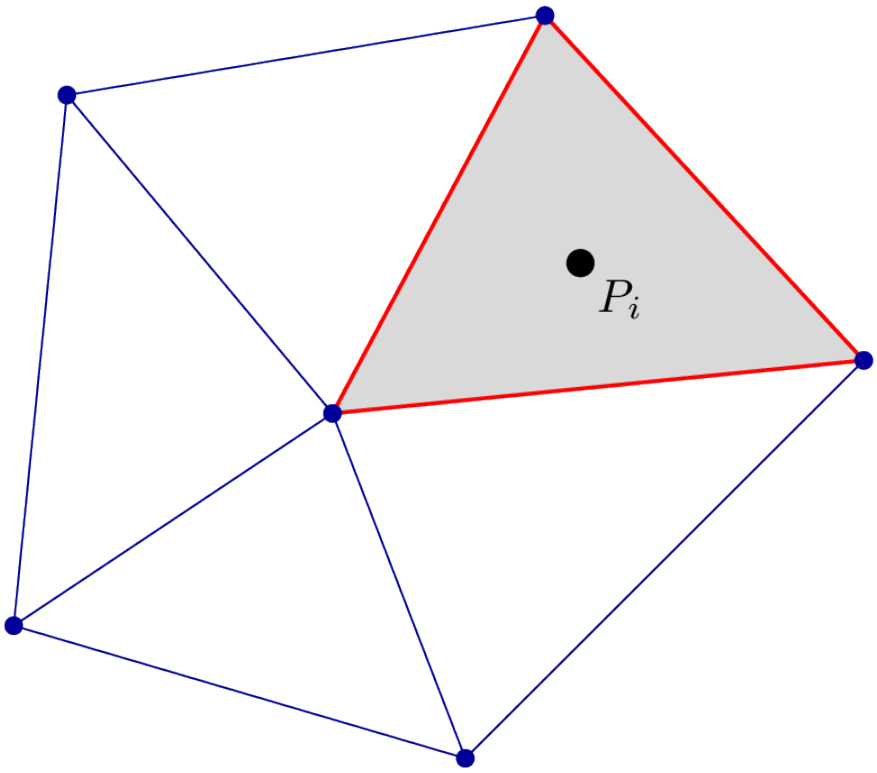

Figure 3: Cells and cell faces

4 Pros and cons

Direct access to cell values, easier to implement, less memory. Harder to handle boundary nodes, less robust on certain meshes.

4.1 Node-centered

In a vertex-centered approach, we first construct a **Voronoi**-based **dual grid**.

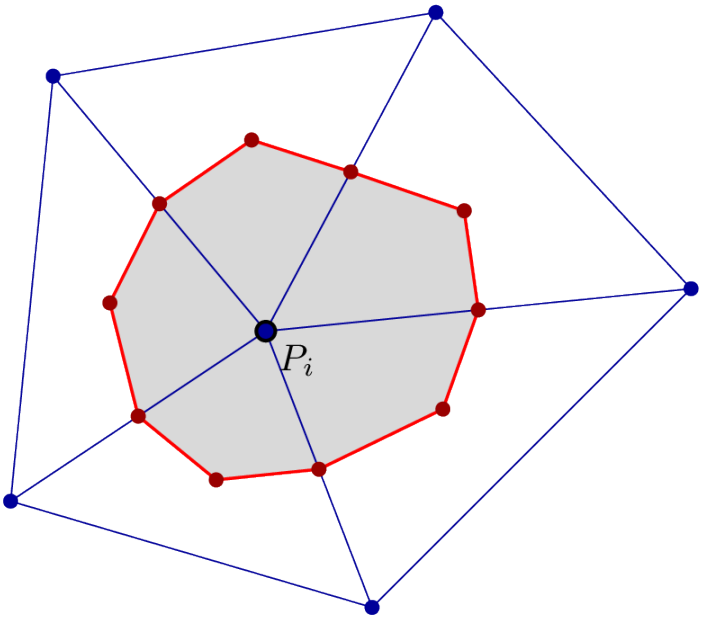

Figure 4: Cells and cell faces

> **i** Pros and cons
>
> More robust on skewed meshes, provides more stable computation of gradients, boundary nodes are easier to handle. Harder to implement, requires more memory.

5 Solver overview

Cell-centered	Vertex-centered
OpenFOAM	SU2
Ansys Fluent	DLR Tau
Eilmer	
DLR Coda	

6 Riemann solvers

We still haven't discussed how we actually compute the flux at the interface.

6.1 Riemann problem

For a 1-D hyperbolic conservation law

$$\partial_t \mathbf{u} + \partial_x \mathbf{f}(\mathbf{u}) = 0$$

define initial conditions

$$\mathbf{u}(x, t=0) = \begin{cases} \mathbf{u}_L & x < 0 \\ \mathbf{u}_R & x > 0 \end{cases}$$

This is called the Riemann problem, we denote the solution as $\mathbf{u}^{\mathcal{RP}}(x,t)$.

> **!** Self-similarity
>
> Solutions to the Riemann problem are self-similar, i.e $\mathbf{u}^{\mathcal{RP}}(x,t) = const$ along $x/t = const$. So solution is only dependent on x/t; this also means that **for any** $t > 0$, $\mathbf{u}^{\mathcal{RP}}(0,t) \equiv const$.

6.2 Example: linear advection

Simplest textbook example:

$$\partial_t u + a\partial_x u = 0.$$

Solution to Riemann problem is given by

$$u^{\mathcal{RP}}(x,t) = \begin{cases} u_L & x/t < a, \\ u_R & x/t > a. \end{cases}$$

In fact, any system of linear transport equations

$$\partial_t \mathbf{u} + \mathbf{A}\partial_x \mathbf{u} = 0.$$

has a simple linear solution related to the eigenvalues and eigenvectors of $\mathbf{A}$.

6.3 Godunov's method

Take our conservation law (in 1D for simplicity)

$$\partial_t \int_{x_{i-1/2}}^{x_{i+1/2}} \mathbf{u}\,\mathrm{d}x + \int_{x_{i-1/2}}^{x_{i+1/2}} \partial_x \mathbf{f}(\mathbf{u})\mathrm{d}x = 0,$$

and integrate from time t^n to t^{n+1}:

$$\int_{t^n}^{t^{n+1}} \partial_t \int_{x_{i-1/2}}^{x_{i+1/2}} \mathbf{u}\,\mathrm{d}x\mathrm{d}t + \int_{t^n}^{t^{n+1}} \int_{x_{i-1/2}}^{x_{i+1/2}} \partial_x \mathbf{f}(\mathbf{u})\mathrm{d}x\mathrm{d}t = 0 \implies$$

$$\frac{\mathbf{u}_i^{n+1} - \mathbf{u}_i^n}{\Delta t} + \frac{1}{\Delta x}\int_{t^n}^{t^{n+1}} (\mathbf{f}(\mathbf{u}_{i+1/2}) - \mathbf{f}(\mathbf{u}_{i-1/2}))\mathrm{d}t = 0.$$

Question

How do we 1) compute flux at $x = i \pm 1/2$ (solution is discontinuous across cell interface) and 2) compute time integral of flux difference?

6.4 Godunov's method

If we take $\mathbf{u} \equiv const$ in each cell, then we have

1. Riemann problem at each interface
2. Solution of Riemann problem at $x/t = 0$ stays constant $\implies$ flux of Riemann problem solution also stays constant in time

So we can write

$$\int_{t^n}^{t^{n+1}} (\mathbf{f}(\mathbf{u}_{i+1/2}) - \mathbf{f}(\mathbf{u}_{i-1/2}))\mathrm{d}t = \Delta t(\mathbf{f}(u^{\mathcal{RP}}_{i+1/2}) - \mathbf{f}(u^{\mathcal{RP}}_{i-1/2}))$$

CFL restriction

Immediately see that we have a restriction $\lambda\Delta t/\Delta x < \frac{1}{2}$, where λ is fastest wavespeed in Riemann problem: i.e. Riemann problems from neighbouring cells do not interact within the timestep.

Question

What are the issues with Godunov's method?

6.5 The birth of modern(-er) CFD

Loss of self-similarity

If we use non-constant values within the cell, the solution to the Riemann problem is no longer self-similar.

- Bram van Leer (1979): we can use a linear reconstruction within the cell to achieve 2-nd order accuracy
- Philip Roe (1981): we don't need exact solution of the Riemann problem

6.6 Roe's method

The key insight is: better to solve Riemann problem approximately and gain solution accuracy **elsewhere**. Which approximation? Local linearization (at cell interface):

$$\partial_t \mathbf{u} + \nabla \mathbf{f}(\mathbf{u}) = 0 \implies \partial_t \mathbf{u} + \mathbf{A}(\mathbf{u}_L, \mathbf{u}_R)\nabla \mathbf{u} = 0.$$

Conditions on $\mathbf{A}$ (**Roe matrix**):

1. $\mathbf{A}$ is diagonalizable with real eigenvalues (remain hyperbolic)
2. $\mathbf{A}$ is consistent: $\mathbf{A}(\mathbf{u}, \mathbf{u}) = \mathbf{A}(\mathbf{u})$

3. $\mathbf{f}(\mathbf{u}_R) - \mathbf{f}(\mathbf{u}_L) = \mathbf{A}(\mathbf{u}_L, \mathbf{u}_R)(\mathbf{u}_R - \mathbf{u}_L)$

One way to define this matrix is as $\mathbf{A} = \frac{\partial \mathbf{f}}{\partial \mathbf{u}}\left(\mathbf{u}_{avg}\right)$, where $\mathbf{u}_{avg}$ is **some** average of the states $\mathbf{u}_L$ and $\mathbf{u}_R$.

6.7 Roe's method

Recipe:

1. Compute $\mathbf{u}_{avg}$, $\mathbf{A}$
2. $\mathbf{A}$ is diagonalizable, i.e. $\mathbf{A} = \mathbf{R}\mathbf{R}^{-1}$
3. Compute $|\mathbf{A}| := \mathbf{R}|\mathbf{R}^{-1}$

For Euler (1D):

- $\rho_{Roe} = \sqrt{\rho_L \rho_R}$
- $u_{Roe} = \dfrac{\sqrt{\rho_L}u_L + \sqrt{\rho_R}u_R}{\sqrt{\rho_L} + \sqrt{\rho_R}}$
- $h_{Roe} = \dfrac{\sqrt{\rho_L}h_L + \sqrt{\rho_R}h_R}{\sqrt{\rho_L} + \sqrt{\rho_R}}$ $(h = e + p/\rho)$

6.8 Roe's method: summary

- Computationally expensive
- Not always stable (hypersonic flows):
  - if $f'(u_L) < 0 < f'(u_R)$ (rarefaction) leads to unphysical solution
  - "Carbuncle phenomena"

6.9 Other Riemann solvers

- HLL (Harten, Lax, Leer): assumes we know two fastest wave speeds of Riemann problem, resolves only them: need *wavespeed estimates*, i.e. min/max eigenvalues of $\mathbf{A}$
  - cannot resolve contact waves
- HLLC
  - add contact wave (need speed estimate, etc.)
- Hybrid methods (Roe + HLL)
- AUSM, AUSM+
  - high-speed flows, splitting of transport due to pressure and convection
  - efficient (no eigendecomposition required)

7 Reconstruction and limiting

7.1 TVD and Godunov's theorem

Godunov's theorem

A linear, monotonicity-preserving method can be at most first-order accurate.

Question

Is all lost?

7.2 MUSCL

- Reconstruct piecewise linear solution from neighbouring cell averages
- Use values at cell interfaces to compute fluxes

$$\mathbf{u}_L = \mathbf{u}_i + \nabla \mathbf{u}_i \cdot (\mathbf{r}_m - \mathbf{r}_i), \quad \mathbf{u}_R = \mathbf{u}_j + \nabla \mathbf{u}_j \cdot (\mathbf{r}_m - \mathbf{r}_j)$$

- Gradient computed either based on linear interpolation or least-squares

Question

We use a Riemann solver and don't account for the gradient of the solution in the cell. Why do we still gain accuracy?

Taylor series expansion analysis

For a smooth scalar field u:

$$u(x_{i+1/2}) = u_i + \left.\frac{\partial u}{\partial x}\right|_i \Delta x + \frac{1}{2}\left.\frac{\partial^2 u}{\partial x^2}\right|_i \Delta x^2 + \ldots$$

Cell-wise constant values disregard the gradient and higher-order terms; Riemann solver adds diffusion proportional to jump at interface - so even though we omit the gradient information within the Riemann solver, we gain accuracy in the other components of the method.

7.3 Slope limiting

Replace $\mathbf{u}_i + \nabla \mathbf{u}_i \cdot (\mathbf{r}_m - \mathbf{r}_i)$ with $\mathbf{u}_i + \Psi \nabla \mathbf{u}_i \cdot (\mathbf{r}_m - \mathbf{r}_i)$.

$\Psi = 0$ recovers 1st order scheme, Ψ - **slope limiter**. Restricts local solution slope to avoid oscillations.

Structured vs unstructured limiters

On structured grids (SG), limiter is function of local slope in specific direction. On unstructured grids (UG), we need to consider all directions and apply **strongest** limiter. Many more limiters for SG (minmod, Superbee, van Leer, van Albada) than UG (Barth-Jespersen, Venkatakrishnan, Nishikawa).

Flux limiting

Alternative approach is to limit values of fluxes $\mathbf{g}_{ij}$ (and not values of $\mathbf{u}$ used to compute fluxes). Flux limiting is harder to construct in multidimensional cases/on unstructured grids, but can be shown to be equivalent to slope limiting.

8 Examples

Online 1D Euler solver

Github

9 OpenFOAM

`rhoCentralFoam` - compressible Euler/Navier–Stokes equations.

Some options:

- `ddtSchemes`: `Euler`, `backward`, `CrankNicolson`
- `gradSchemes`, `divSchemes`, `interpolationSchemes`: `Gauss/leastSquares`, `faceMDLimited/cellMDLimite`
  `linear/vanLeer/limitedLinear` (minmod limiter):
- `fluxScheme`: `Kurganov`, `HLLC`, `HLL`

```
gradSchemes
{
    default cellMDLimited Gauss linear 1;
}
```

10 References

- R. LeVeque, Numerical Methods for Conservation Laws
- J. Quirk, A contribution to the great Riemann solver debate
- H. Nishikawa et al., Very simple, carbuncle-free, boundary-layer-resolving, rotated-hybrid Riemann solvers, J. Comp. Phys.