DUNE PDELab Tutorial 07 - Overview

Discontinuous Galerkin for Hyperbolic conservation laws

Linus Seelinger

Interdisziplinäres Zentrum für Wissenschaftliches Rechnen
Im Neuenheimer Feld 205, D-69120 Heidelberg

August 31, 2021

Motivation

Discontinuous Galerkin methods offer

- local mass conservation (Conforming FEM does not!)
- higher-order approximation (Finite Volume does not!)

First-Order Hyperbolic PDEs

Conservative Form

$$\partial_t u(x,t) + \nabla \cdot F(u(x,t),x,t) = g(u(x,t),x,t) \quad \text{in } U = \Omega \times \Sigma$$

with initial conditions

$$u(x,0) = u_0(x)$$

- Spatial domain $\Omega = \mathbb{R}^d$, $d \in \mathbb{N}$, temporal domain $\Sigma = \mathbb{R}^+$
- $F(u,x,t)$ is called *flux function*
- Arises naturally from conservation laws (e.g. mass, energy, ...)

Quasi-Linear Form

$$\partial_t u(x,t)+\sum_{j=1}^{d} B_j(u(x,t),x,t)\partial_{x_j} u(x,t)+\tilde{g}(u(x,t),x,t)=0 \quad \text{in } \Omega\times\Sigma$$

with initial conditions

$$u(x,0)=u_0(x)$$

- Derived from conservative form via chain rule (needs smoothness of flux function)

Hyperbolicity Criterion

$$\partial_t u(x,t)+\sum_{j=1}^{d} B_j(u(x,t),x,t)\partial_{x_j} u(x,t)+\tilde{g}(u(x,t),x,t)=0 \quad \text{in } \Omega \times \Sigma$$

- Is called *hyperbolic* if

  $$B(u,x,t;y)=\sum_{j=1}^{d} y_j B_j(u,x,t)$$

  is real diagonalizable (i.e. real eigenvalues, eigenvectors form basis)
- Property needed for theoretical and numerical treatment

Examples

Acoustic Wave Equation

Linearized conservation laws:

$$
\begin{aligned}
\partial_t \tilde{\rho} + \nabla \cdot (\bar{\rho} \tilde{v}) = 0 \qquad & \text{(conservation of mass)} \\
\partial_t (\bar{\rho} \tilde{v}) + \nabla \tilde{p} = 0 \qquad & \text{(conservation of momentum)}
\end{aligned}
$$

- Describes propagation of acoustic waves through material
- Derived by linearizing mass/momentum conservation around background state
- Simplifying assumptions involved!

Acoustic Wave Equation

As hyperbolic system:

$$\partial_t u(x,t) + \nabla \cdot F(u(x,t),x,t) = 0,$$

where

$$u = \begin{pmatrix} \varrho \\ q_1 \\ \vdots \\ q_d \end{pmatrix}, \quad F(u(x,t),x,t) = \begin{pmatrix} q_1 & q_2 & \dots & q_d \\ c^2\rho & 0 & \dots & 0 \\ 0 & c^2\rho & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & c^2\rho \end{pmatrix} \in \mathbb{R}^{m\times d}$$

- Derived replacing momentum by $\tilde{q} = \bar{\rho}\tilde{v}$

Shallow Water Equations

$$\partial_t \begin{pmatrix} h \\ u_1 h \\ u_2 h \end{pmatrix} + \nabla \cdot \begin{pmatrix} u_1 h & u_2 h \\ u_1^2 h + \frac{1}{2} g h^2 & u_1 u_2 h \\ u_1 u_2 h & u_2^2 h + \frac{1}{2} g h^2 \end{pmatrix} = 0$$

- Describes water height $h > 0$ and velocity $u = (u_1, u_2)$
- Widely used for predictions of flooding, dam-breaks, tsunamis
- Nonlinear PDE, again derived from conservation law

Discontinuous Galerkin Methods

Finite Element Space

$$V_h^q = \left\{ v \in L^2(\Omega) \,:\, v|_e = p \circ \mu_e^{-1}, p \in \mathbb{P}^{q,d} \right\}$$

- $\mu_e : \hat{E} \to e$ maps reference element to mesh element
- $p \in \mathbb{P}^{q,d}$ polynomial space on ref. element
- Note: No continuity required $\to$ nonconforming method!

Discretization

For piecewise smooth test function v:

$$\int_\Omega \left[\partial_t u + \sum_{j=1}^{d} \partial_{x_j} F_j(u,x,t)\right] \cdot v \, dx =$$

$$= d_t(u,v)_\Omega - \sum_{e \in \mathcal{E}_h} \int_e F(u,x,t) : \nabla v \, dx$$

$$+ \sum_{f \in \mathcal{F}_h^i} \int_f [\![(F(u,s,t)n) \cdot v]\!] \, ds + \sum_{f \in \mathcal{F}_h^{\partial\Omega}} \int_f (F(u,s,t)n) \cdot v \, ds \,.$$

- $[\![v]\!](x) = v^-(x) - v^+(x)$ denotes jump across interface
- Need single-valued approximation of $F \cdot n$

Numerical Fluxes

$$\Phi : \mathbb{R}^d \times U \times U \to \mathbb{R},$$

approximating $F \cdot n$ on interfaces, fulfilling

- Consistency: Approximation exact if solution locally continuous
- Conservation: $\Phi(n_1, u_1, u_2) + \Phi(n_2, u_2, u_1) = 0$

Implementation in DUNE

Implementation in DUNE

- Define particular numerical flux
- Define DG method's spatial and temporal local operators
- Use a `DGLocalFiniteElementMap`

Otherwise, same as any other instationary method!