Entropy stable high order discontinuous Galerkin methods for nonlinear conservation laws

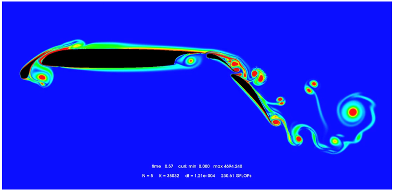
Jesse Chan

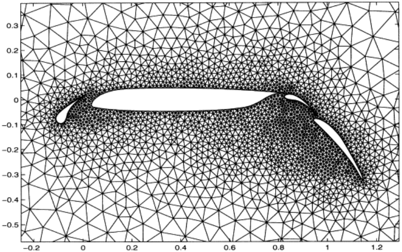
[1]Department of Computational and Applied Mathematics

Department of Mathematics, Purdue University
September 24, 2018

High order finite element methods for hyperbolic PDEs

- Focus: high accuracy computational mechanics on complex geometries.
- Applications in fluid dynamics (waves, vorticular flows, turbulence, shocks).
- High order approximations are more accurate per unknown.
- High performance computing on many-core architectures (efficient explicit time-stepping).

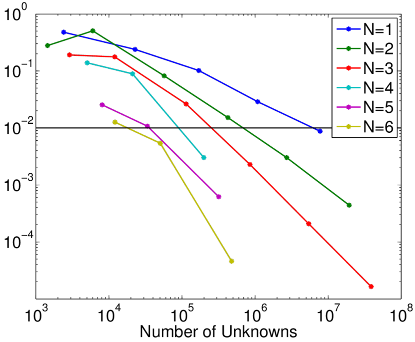




Mesh from Slawig 2001.

High order finite element methods for hyperbolic PDEs

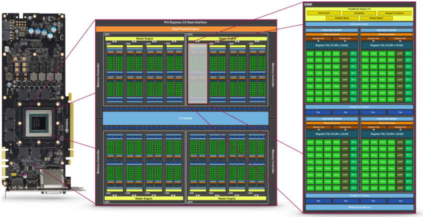
- Focus: high accuracy computational mechanics on complex geometries.
- Applications in fluid dynamics (waves, vorticular flows, turbulence, shocks).
- High order approximations are more accurate per unknown.
- High performance computing on many-core architectures (efficient explicit time-stepping).



For smooth solutions, high order methods deliver a lower error per degree of freedom.

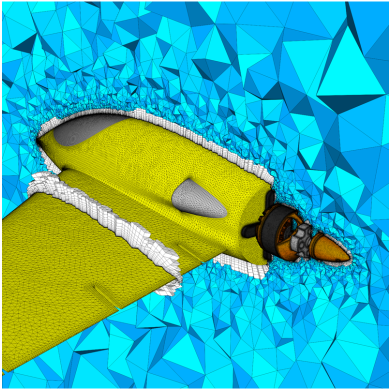
High order finite element methods for hyperbolic PDEs

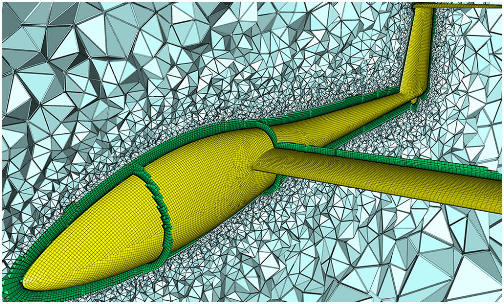
- Focus: high accuracy computational mechanics on complex geometries.
- Applications in fluid dynamics (waves, vorticular flows, turbulence, shocks).
- High order approximations are more accurate per unknown.
- High performance computing on many-core architectures (efficient explicit time-stepping).



Schematic of an NVIDIA graphics processing unit (GPU).

Finite element methods: general unstructured meshes

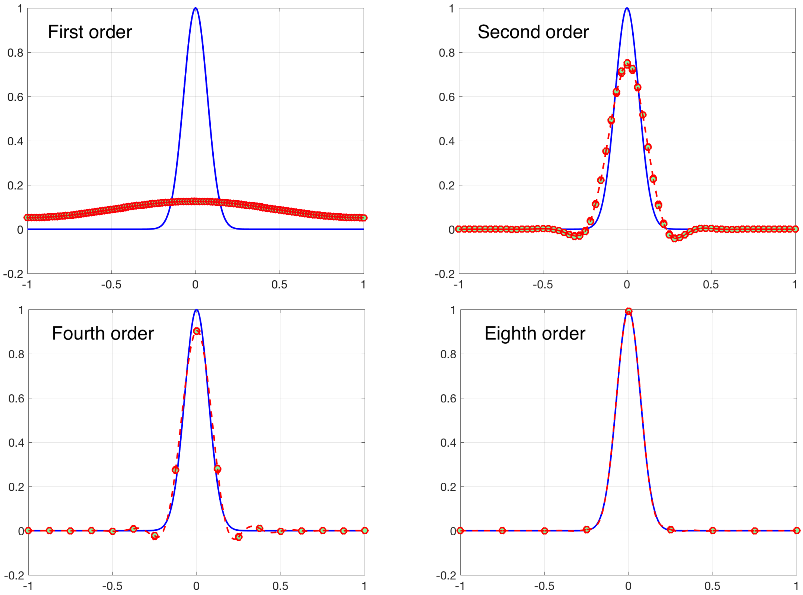




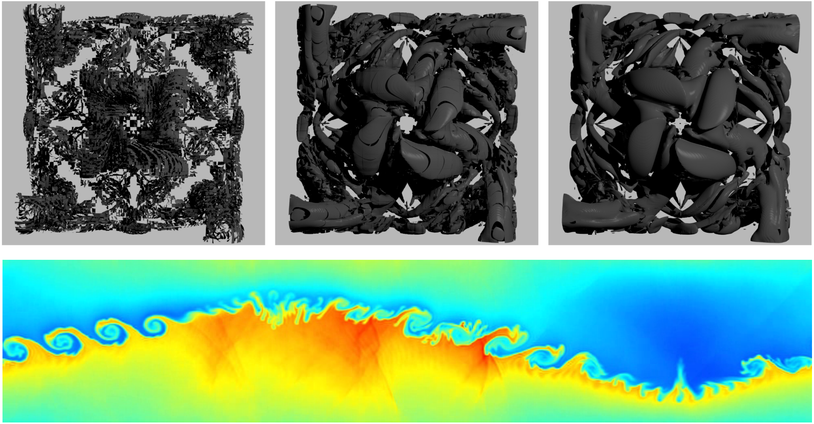
DG methods are compatible with unstructured meshes containing different types of elements (tetrahedra, hexahedra most common, but also prisms and pyramids).

Figures courtesy of Pointwise Inc (https://www.pointwise.com).

High order decreases numerical dissipation



High order decreases numerical dissipation



2nd, 4th, and 16th order Taylor-Green (top), 8th order Kelvin-Helmholtz (bottom).

Beck, Gassner (2012). *Numerical Simulation of the Taylor-Green Vortex at Re=1600 with the Discontinuous Galerkin Spectral Element Method for well-resolved and underresolved scenarios*

Peraire, Persson (2010). *High-Order Discontinuous Galerkin Methods for CFD*

Talk outline

1. Stability of high order DG: linear vs nonlinear PDEs
2. Summation by parts finite differences and high order DG
3. Entropy stable formulations and flux differencing
4. Numerical experiments
   - 1D experiments
   - Triangular and tetrahedral meshes
   - Quadrilateral and hexahedral meshes

Talk outline

1 Stability of high order DG: linear vs nonlinear PDEs

2 Summation by parts finite differences and high order DG

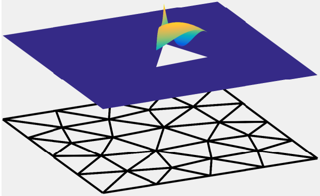
3 Entropy stable formulations and flux differencing

4 Numerical experiments
- 1D experiments
- Triangular and tetrahedral meshes
- Quadrilateral and hexahedral meshes

Basics of discontinuous Galerkin methods

Discontinuous Galerkin (DG) methods:

- High order accuracy, geometric flexibility.
- Weak continuity across faces.



- Continuous PDE (example: advection)

$$\frac{\partial u}{\partial t} = \frac{\partial f(u)}{\partial x}, \qquad f(u) = u.$$

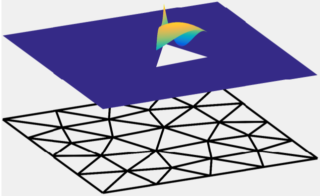
- Local DG form with numerical flux $\boldsymbol{f}^*$: find $u \in P^N\left(D^k\right)$ such that

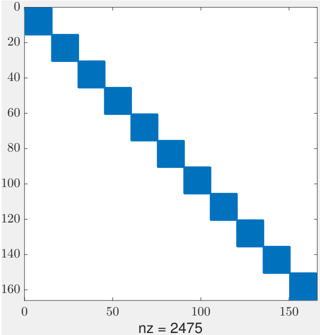
$$\int_{D_k} \frac{\partial u}{\partial t}\phi = \int_{D_k} \frac{\partial f(u)}{\partial x}\phi + \int_{\partial D_k} \boldsymbol{n} \cdot \left(\boldsymbol{f}^* - \boldsymbol{f}(u)\right)\phi, \qquad \forall \phi \in P^N\left(D^k\right).$$

Basics of discontinuous Galerkin methods

Discontinuous Galerkin (DG) methods:

- High order accuracy, geometric flexibility.
- Weak continuity across faces.





DG in space yields system of ODEs

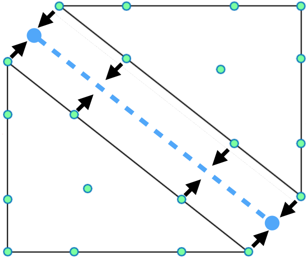
$$\mathbf{M}_{\Omega}\frac{\mathrm{d}\mathbf{u}}{\mathrm{dt}} = \mathbf{A}\mathbf{u}.$$

DG mass matrix decouples across elements, inter-element coupling only through $\mathbf{A}$.

Implementation of explicit time-domain DG methods

Given initial condition $u(\mathbf{x}, 0)$:

- Compute numerical flux on element faces (non-local).
- Compute RHS of (local) ODE.
- Evolve (local) solution using explicit time integration (RK, AB, etc).

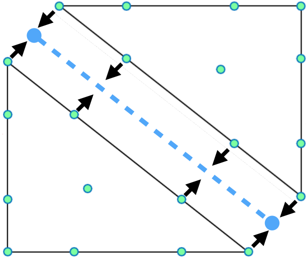


$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{dt}} = \mathbf{D}_x \mathbf{u} + \sum_{\text{faces}} \mathbf{L}_f \left(\text{flux}\right), \qquad \mathbf{L}_f = \mathbf{M}^{-1}\mathbf{M}_f.$$

Implementation of explicit time-domain DG methods

Given initial condition $u(\mathbf{x}, 0)$:

- Compute numerical flux on element faces (non-local).
- Compute RHS of (local) ODE.
- Evolve (local) solution using explicit time integration (RK, AB, etc).

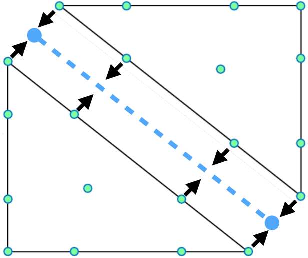


$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = \underbrace{\mathbf{D}_x\mathbf{u}}_{\text{Volume}} + \underbrace{\sum_{\text{faces}} \mathbf{L}_f\,(\text{flux})}_{\text{Surface}}, \qquad \mathbf{L}_f = \mathbf{M}^{-1}\mathbf{M}_f.$$

Implementation of explicit time-domain DG methods

Given initial condition $u(\mathbf{x}, 0)$:

- Compute numerical flux on element faces (non-local).
- Compute RHS of (local) ODE.
- Evolve (local) solution using explicit time integration (RK, AB, etc).

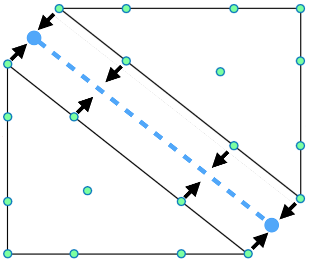


$$\underbrace{\frac{\mathrm{d}\mathbf{u}}{\mathrm{dt}}}_{\text{Update}} = \underbrace{\mathbf{D}_x\mathbf{u}}_{\text{Volume}} + \underbrace{\sum_{\text{faces}} \mathbf{L}_f\,(\text{flux})}_{\text{Surface}}, \qquad \mathbf{L}_f = \mathbf{M}^{-1}\mathbf{M}_f.$$

Implementation of explicit time-domain DG methods

Given initial condition $u(\mathbf{x}, 0)$:

- Compute numerical flux on element faces (non-local).
- Compute RHS of (local) ODE.
- Evolve (local) solution using explicit time integration (RK, AB, etc).



Pros: simple, scalable, and efficient matrix-free implementation.

Cons: explicit time-stepping, high order methods prone to instability. Regularization (slope limiting, artificial viscosity) to avoid blow up!

Must ensure semi-discrete system is inherently *energy stable*!

DG is semi-discretely energy stable for linear advection

- Linear periodic advection on $[-1,1]$

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0, \qquad u(-1) = u(1), \qquad \Longrightarrow \frac{\partial}{\partial t}\|u\|^2_{L^2([-1,1])} = 0.$$

- Triangulate domain with elements D^k, define $[\![u]\!] = u^+ - u$ on D^k.
- DG formulation: find $u(x) \in P^N(D^k)$ s.t. $\forall v \in P^N(D^k)$

$$\sum_k \int_{D^k} \left(\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x}\right) v \, \mathrm{d}x + \frac{1}{2}\int_{\partial D^k} \left([\![u]\!] n_x + \tau [\![u]\!]\right) v \, \mathrm{d}x = 0.$$

- Energy estimate: take $v = u$, chain rule in time, integrate by parts.

$$\sum_k \frac{\partial}{\partial t}\|u\|^2_{D^k} \leq -\sum_k \frac{\tau}{2}\int_{\partial D^k} [\![u]\!]^2 \, \mathrm{d}x.$$

DG is semi-discretely energy stable for linear advection

- Linear periodic advection on $[-1, 1]$

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0, \qquad u(-1) = u(1), \qquad \Longrightarrow \frac{\partial}{\partial t} \|u\|^2_{L^2([-1,1])} = 0.$$

- Triangulate domain with elements D^k, define $[\![u]\!] = u^+ - u$ on D^k.
- DG formulation: find $u(x) \in P^N(D^k)$ s.t. $\forall v \in P^N(D^k)$

$$\sum_k \int_{D^k} \left(\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} \right) v \, \mathrm{d}x + \frac{1}{2} \int_{\partial D^k} \left([\![u]\!] n_x + \tau [\![u]\!] \right) v \, \mathrm{d}x = 0.$$

- Energy estimate: take $v = u$, chain rule in time, integrate by parts.

$$\sum_k \frac{\partial}{\partial t} \|u\|^2_{D^k} \leq -\sum_k \frac{\tau}{2} \int_{\partial D^k} [\![u]\!]^2 \, \mathrm{d}x.$$

DG is semi-discretely energy stable for linear advection

- Linear periodic advection on $[-1,1]$

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0, \qquad u(-1) = u(1), \qquad \Longrightarrow \frac{\partial}{\partial t} \|u\|^2_{L^2([-1,1])} = 0.$$

- Triangulate domain with elements D^k, define $[\![u]\!] = u^+ - u$ on D^k.
- DG formulation: find $u(x) \in P^N(D^k)$ s.t. $\forall v \in P^N(D^k)$

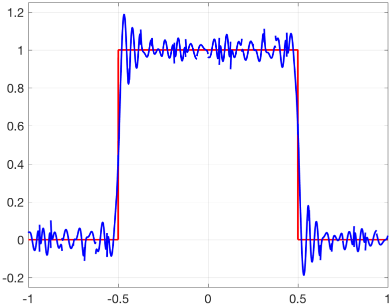
$$\sum_k \int_{D^k} \left(\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} \right) v \, \mathrm{d}x + \frac{1}{2} \int_{\partial D^k} \left([\![u]\!] n_x + \tau [\![u]\!] \right) v \, \mathrm{d}x = 0.$$

- Energy estimate: take $v = u$, chain rule in time, integrate by parts.

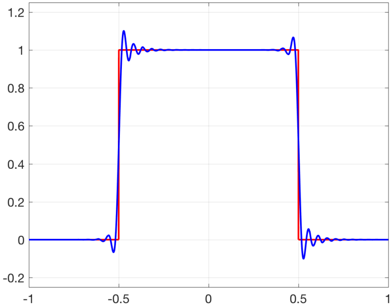
$$\sum_k \frac{\partial}{\partial t} \|u\|^2_{D^k} \leq -\sum_k \frac{\tau}{2} \int_{\partial D^k} [\![u]\!]^2 \, \mathrm{d}x.$$

Energy conservative vs. energy stable DG methods

- Energy estimate: implies solution is non-increasing if $\tau \geq 0$.
- Energy conservative (non-dissipative) "central" flux when $\tau = 0$.
- Energy stable (dissipative) "Lax-Friedrichs" flux when $\tau = 1$.



(a) Energy conservative ($\tau = 0$)



(b) Energy stable ($\tau = 1$)

Generalization to nonlinear problems: entropy stability

- Generalizes energy stability to nonlinear systems of conservation laws (Burgers', shallow water, compressible Euler, MHD).

$$\frac{\partial \boldsymbol{u}}{\partial t} + \frac{\partial \boldsymbol{f}(\boldsymbol{u})}{\partial x} = 0.$$

- Continuous entropy inequality: given a convex entropy function $S(\boldsymbol{u})$ and "entropy potential" $\psi(\boldsymbol{u})$,

$$\int_\Omega \boldsymbol{v}^T \left(\frac{\partial \boldsymbol{u}}{\partial t} + \frac{\partial \boldsymbol{f}(\boldsymbol{u})}{\partial x}\right) = 0, \qquad \boldsymbol{v} = \frac{\partial S}{\partial \boldsymbol{u}}$$
$$\Longrightarrow \int_\Omega \frac{\partial S(\boldsymbol{u})}{\partial t} + \left(\boldsymbol{v}^T \boldsymbol{f}(\boldsymbol{u}) - \psi(\boldsymbol{u})\right)\Big|_{-1}^{1} \leq 0.$$

- Proof of entropy inequality relies on chain rule, integration by parts.

Example: compressible flow and mathematical entropy

- Conservative variables: density, momentum, energy

$$\boldsymbol{u} = (\rho, \boldsymbol{m}, E), \qquad \rho > 0, \qquad E > \frac{1}{2}|\boldsymbol{m}|^2/\rho.$$

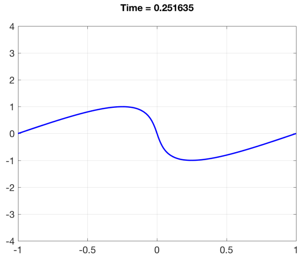
- Physical entropy $s(\boldsymbol{u})$ always increasing; mathematical entropy $S(\boldsymbol{u})$ always decreasing (analogous to energy).

$$s(\boldsymbol{u}) = \log\left(\frac{(\gamma-1)\rho e}{\rho^\gamma}\right), \qquad S(\boldsymbol{u}) = -\rho s(\boldsymbol{u}).$$

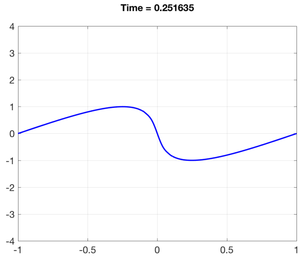
- Entropy variables $\boldsymbol{v}(\boldsymbol{u})$: invertible function of $\boldsymbol{u}$

$$\boldsymbol{v}(\boldsymbol{u}) = \frac{\partial S}{\partial \boldsymbol{u}} = \frac{1}{\rho e}\begin{pmatrix} \rho e(\gamma + 1 - s(\boldsymbol{u})) - E \\ m \\ -\rho \end{pmatrix}$$

Why are discretizations of nonlinear PDEs unstable?



(a) $N = 7, K = 8$ (aligned mesh)



(b) $N = 7, K = 9$ (non-aligned mesh)

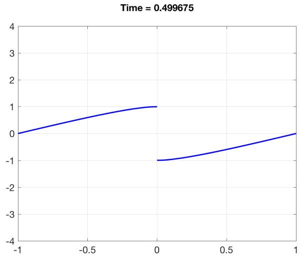
- Burgers' equation: $f(u) = u^2/2$. How to compute $\frac{\partial}{\partial x} f(u)$?

$$\frac{\partial u}{\partial t} + \frac{1}{2}\frac{\partial u^2}{\partial x} = 0, \qquad u \in P^N(D^k), \quad u^2 \notin P^N(D^k).$$

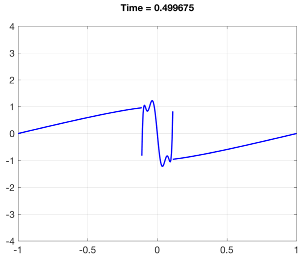
- Differentiating L^2 projection P_N + inexact quadrature: no chain rule.

$$\int_{D^k} \left(\frac{\partial u}{\partial t} + \frac{1}{2}\frac{\partial}{\partial x} P_N u^2\right) v \,\mathrm{d}x = 0, \qquad \frac{1}{2}\frac{\partial P_N u^2}{\partial x} \neq P_N\left(u\frac{\partial u}{\partial x}\right)$$

Why are discretizations of nonlinear PDEs unstable?



(a) $N = 7, K = 8$ (aligned mesh)



(b) $N = 7, K = 9$ (non-aligned mesh)

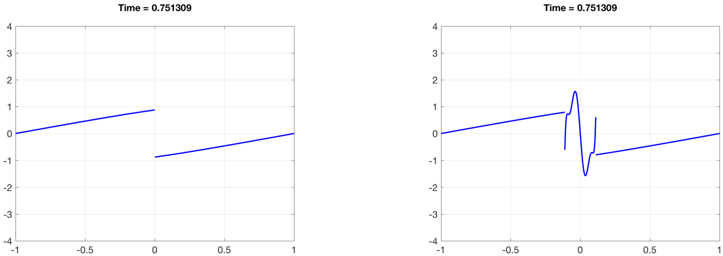
- Burgers' equation: $f(u) = u^2/2$. How to compute $\frac{\partial}{\partial x} f(u)$?

$$\frac{\partial u}{\partial t} + \frac{1}{2}\frac{\partial u^2}{\partial x} = 0, \qquad u \in P^N(D^k), \quad u^2 \notin P^N(D^k).$$

- Differentiating L^2 projection P_N + inexact quadrature: no chain rule.

$$\int_{D^k} \left(\frac{\partial u}{\partial t} + \frac{1}{2}\frac{\partial}{\partial x} P_N u^2 \right) v \, \mathrm{d}x = 0, \qquad \frac{1}{2}\frac{\partial P_N u^2}{\partial x} \neq P_N \left(u \frac{\partial u}{\partial x} \right)$$

Why are discretizations of nonlinear PDEs unstable?



(a) $N = 7, K = 8$ (aligned mesh) (b) $N = 7, K = 9$ (non-aligned mesh)

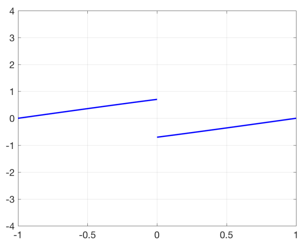
- Burgers' equation: $f(u) = u^2/2$. How to compute $\frac{\partial}{\partial x} f(u)$?

$$\frac{\partial u}{\partial t} + \frac{1}{2}\frac{\partial u^2}{\partial x} = 0, \qquad u \in P^N(D^k), \quad u^2 \notin P^N(D^k).$$

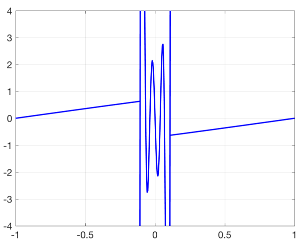
- Differentiating L^2 projection P_N + inexact quadrature: no chain rule.

$$\int_{D^k} \left(\frac{\partial u}{\partial t} + \frac{1}{2}\frac{\partial}{\partial x} P_N u^2 \right) v \, \mathrm{d}x = 0, \qquad \frac{1}{2}\frac{\partial P_N u^2}{\partial x} \neq P_N \left(u \frac{\partial u}{\partial x} \right)$$

Why are discretizations of nonlinear PDEs unstable?



(a) $N = 7, K = 8$ (aligned mesh)



(b) $N = 7, K = 9$ (non-aligned mesh)

- Burgers' equation: $f(u) = u^2/2$. How to compute $\frac{\partial}{\partial x} f(u)$?

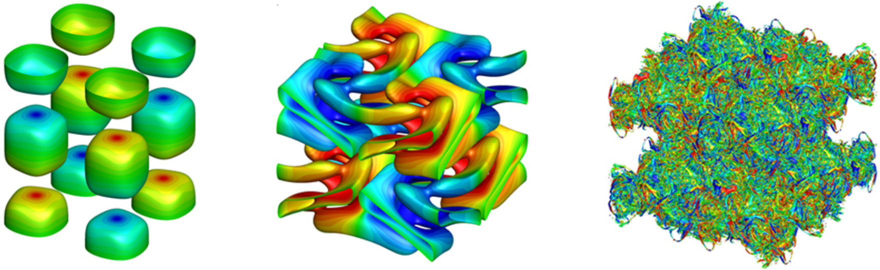
$$\frac{\partial u}{\partial t} + \frac{1}{2}\frac{\partial u^2}{\partial x} = 0, \qquad u \in P^N(D^k), \quad u^2 \notin P^N(D^k).$$

- Differentiating L^2 projection P_N + inexact quadrature: no chain rule.

$$\int_{D^k} \left(\frac{\partial u}{\partial t} + \frac{1}{2}\frac{\partial}{\partial x} P_N u^2 \right) v \, \mathrm{d}x = 0, \qquad \frac{1}{2}\frac{\partial P_N u^2}{\partial x} \neq P_N \left(u \frac{\partial u}{\partial x} \right)$$

Tradeoff: high order accuracy vs stability

- Asymptotic stability for smooth solutions (not shocks or turbulence!)
- Common fix: stabilize by regularizing (limiters, filters, art. viscosity).

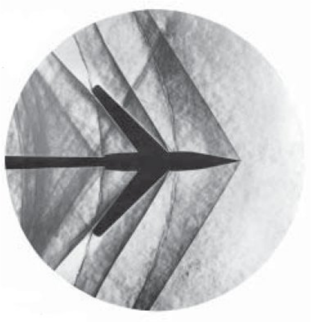


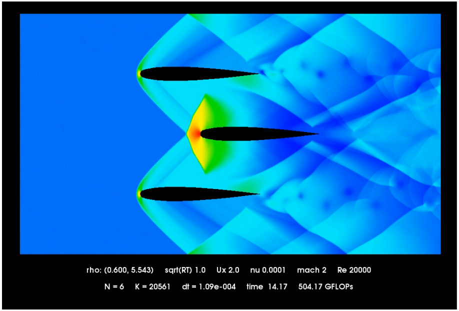
Under-resolved solutions: turbulence (inviscid Taylor-Green vortex).

Figures courtesy of Gregor Gassner, T. Warburton, Coastal Inlets Research Program (CIRP), "Man on Wire" (2008).

Tradeoff: high order accuracy vs stability

- Asymptotic stability for smooth solutions (not shocks or turbulence!)
- Common fix: stabilize by regularizing (limiters, filters, art. viscosity).



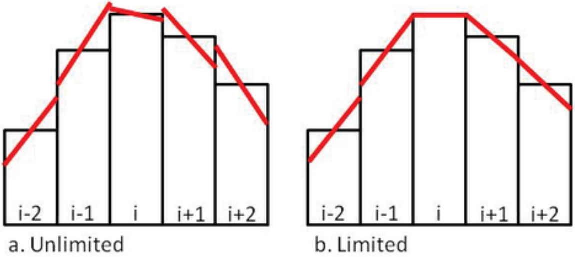


Under-resolved solutions: shock waves.

Figures courtesy of Gregor Gassner, T. Warburton, Coastal Inlets Research Program (CIRP), "Man on Wire" (2008).

Tradeoff: high order accuracy vs stability

- Asymptotic stability for smooth solutions (not shocks or turbulence!)
- Common fix: stabilize by regularizing (limiters, filters, art. viscosity).

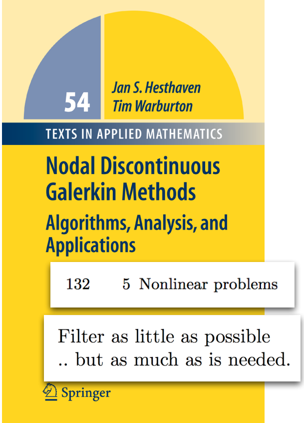


Slope limiting for a finite volume method.

Figures courtesy of Gregor Gassner, T. Warburton, Coastal Inlets Research Program (CIRP), "Man on Wire" (2008).

Tradeoff: high order accuracy vs stability

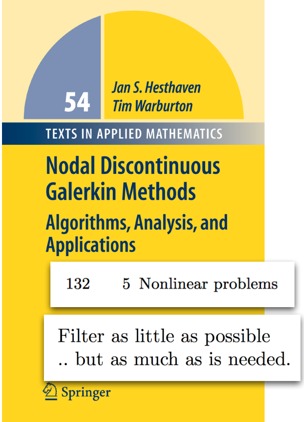
- Asymptotic stability for smooth solutions (not shocks or turbulence!)
- Common fix: stabilize by regularizing (limiters, filters, art. viscosity).

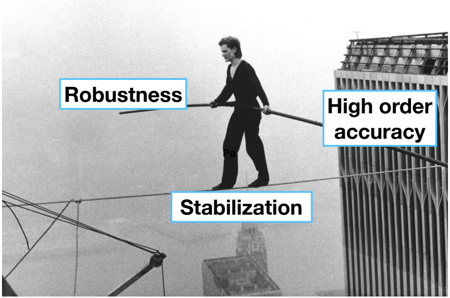


Figures courtesy of Gregor Gassner, T. Warburton, Coastal Inlets Research Program (CIRP), "Man on Wire" (2008).

Tradeoff: high order accuracy vs stability

- Asymptotic stability for smooth solutions (not shocks or turbulence!)
- Common fix: stabilize by regularizing (limiters, filters, art. viscosity).





Figures courtesy of Gregor Gassner, T. Warburton, Coastal Inlets Research Program (CIRP), "Man on Wire" (2008).

Talk outline

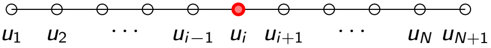
1 Stability of high order DG: linear vs nonlinear PDEs

2 Summation by parts finite differences and high order DG

3 Entropy stable formulations and flux differencing

4 Numerical experiments
- 1D experiments
- Triangular and tetrahedral meshes
- Quadrilateral and hexahedral meshes

Summation-by-parts (SBP) finite differences

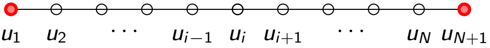


Simplest SBP finite difference matrix: combine 2nd order finite difference formulas at interior points with 1st order finite differences at boundary points .

$$\left.\frac{\partial u}{\partial x}\right|_{x=x_i} \approx \frac{u_{i+1}-u_{i-1}}{2\Delta x} \qquad \text{(at interior points } x_i\text{)},$$

$$\boldsymbol{D} = \frac{1}{2\Delta x}\begin{bmatrix} ? & ? & & \\ -1 & 0 & 1 & \\ & -1 & 0 & \ddots \\ & & \ddots & \ddots \end{bmatrix}, \qquad \boldsymbol{M} = \Delta x \begin{bmatrix} ? & & & \\ & 1 & & \\ & & 1 & \\ & & & \ddots \end{bmatrix}.$$

Summation-by-parts (SBP) finite differences

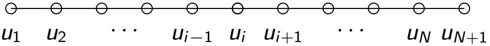


Simplest SBP finite difference matrix: combine 2nd order finite difference formulas at interior points with 1st order finite differences at boundary points .

$$\left.\frac{\partial u}{\partial x}\right|_{x=x_i} \approx \frac{u_2 - u_1}{\Delta x}, \qquad \frac{u_{N+1} - u_N}{\Delta x} \qquad \text{(at boundary pts } x_i\text{)}$$

$$\boldsymbol{D} = \frac{1}{2\Delta x}\begin{bmatrix} -2 & 2 & & \\ -1 & 0 & 1 & \\ & -1 & 0 & \ddots \\ & & \ddots & \ddots \end{bmatrix}, \qquad \boldsymbol{M} = \Delta x \begin{bmatrix} 1/2 & & & \\ & 1 & & \\ & & 1 & \\ & & & \ddots \end{bmatrix}.$$

Summation-by-parts (SBP) finite differences



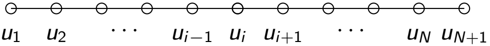
Simplest SBP finite difference matrix: combine 2nd order finite difference formulas at interior points with 1st order finite differences at boundary points .

$$\boldsymbol{MD} = \frac{1}{2}\begin{bmatrix} -1 & 1 & & \\ -1 & 0 & 1 & \\ & -1 & 0 & \ddots \\ & & \ddots & \ddots \end{bmatrix}, \qquad \boldsymbol{MD} + \boldsymbol{D}^T\boldsymbol{M} = \underbrace{\begin{bmatrix} -1 & & & \\ & 0 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}}_{\boldsymbol{B}}.$$

Mimic integration by parts: difference matrix $\boldsymbol{D}$, SPD "norm" matrix $\boldsymbol{M}$

$$\boldsymbol{Q} = \boldsymbol{B} - \boldsymbol{Q}^T, \qquad \boldsymbol{Q} = \boldsymbol{MD}, \qquad \boldsymbol{M} \text{ diagonal.}$$

Summation-by-parts (SBP) finite differences



Simplest SBP finite difference matrix: combine 2nd order finite difference formulas at interior points with 1st order finite differences at boundary points .

$$\boldsymbol{M}\boldsymbol{D} = \frac{1}{2}\begin{bmatrix} -1 & 1 & & \\ -1 & 0 & 1 & \\ & -1 & 0 & \ddots \\ & & \ddots & \ddots \end{bmatrix}, \qquad \boldsymbol{M}\boldsymbol{D} + \boldsymbol{D}^T\boldsymbol{M} = \underbrace{\begin{bmatrix} -1 & & & \\ & 0 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}}_{\boldsymbol{B}}.$$

Mimic integration by parts: difference matrix $\boldsymbol{D}$, SPD "norm" matrix $\boldsymbol{M}$

$$\boldsymbol{Q} = \boldsymbol{B} - \boldsymbol{Q}^T, \qquad \boldsymbol{Q} = \boldsymbol{M}\boldsymbol{D}, \qquad \boldsymbol{M} \text{ diagonal.}$$

Revisiting Burgers' equation: stable split formulations

- Rewrite Burgers' equation in split form

$$\frac{\partial u}{\partial t} + \frac{1}{3}\left(\frac{\partial u^2}{\partial x} + u\frac{\partial u}{\partial x}\right) = 0.$$

- SBP discretization of split formulation

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{dt}} + \frac{1}{3}\left(\boldsymbol{D}\left(\boldsymbol{u}^2\right) + \mathrm{diag}\left(\boldsymbol{u}\right)\boldsymbol{D}\boldsymbol{u}\right) + \boldsymbol{M}^{-1}\boldsymbol{B}\boldsymbol{f}^* = 0, \qquad \boldsymbol{f}^* = \begin{bmatrix} \boldsymbol{f}_0^* \\ \vdots \\ \boldsymbol{f}_N^* \end{bmatrix}.$$

- Semi-discrete stability: multiply by $\boldsymbol{u}^T\boldsymbol{M}$, use $\boldsymbol{Q} = \boldsymbol{M}\boldsymbol{D}$ + diagonal matrices commute + SBP to eliminate volume terms

$$\boldsymbol{u}^T\boldsymbol{M}\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{dt}} + \frac{1}{3}\left(\boldsymbol{u}^T\boldsymbol{Q}\boldsymbol{u}^2 + \boldsymbol{u}^T\boldsymbol{M}\mathrm{diag}\left(\boldsymbol{u}\right)\boldsymbol{D}\boldsymbol{u}\right) + \boldsymbol{u}^T\boldsymbol{B}\boldsymbol{f}^* = 0.$$

Revisiting Burgers' equation: stable split formulations

- Rewrite Burgers' equation in split form

$$\frac{\partial u}{\partial t} + \frac{1}{3}\left(\frac{\partial u^2}{\partial x} + u\frac{\partial u}{\partial x}\right) = 0.$$

- SBP discretization of split formulation

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{dt}} + \frac{1}{3}\left(\boldsymbol{D}\left(\boldsymbol{u}^2\right) + \mathrm{diag}\left(\boldsymbol{u}\right)\boldsymbol{D}\boldsymbol{u}\right) + \boldsymbol{M}^{-1}\boldsymbol{B}\boldsymbol{f}^* = 0, \qquad \boldsymbol{f}^* = \begin{bmatrix} \boldsymbol{f}_0^* \\ \vdots \\ \boldsymbol{f}_N^* \end{bmatrix}.$$

- Semi-discrete stability: multiply by $\boldsymbol{u}^T\boldsymbol{M}$, use $\boldsymbol{Q} = \boldsymbol{M}\boldsymbol{D}$ + diagonal matrices commute + SBP to eliminate volume terms

$$\boldsymbol{u}^T\boldsymbol{M}\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{dt}} + \frac{1}{3}\left(\boldsymbol{u}^T\boldsymbol{Q}\boldsymbol{u}^2 + \boldsymbol{u}^T\boldsymbol{M}\mathrm{diag}\left(\boldsymbol{u}\right)\boldsymbol{D}\boldsymbol{u}\right) + \boldsymbol{u}^T\boldsymbol{B}\boldsymbol{f}^* = 0.$$

Revisiting Burgers' equation: stable split formulations

- Rewrite Burgers' equation in split form

$$\frac{\partial u}{\partial t} + \frac{1}{3}\left(\frac{\partial u^2}{\partial x} + u\frac{\partial u}{\partial x}\right) = 0.$$

- SBP discretization of split formulation

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{dt}} + \frac{1}{3}\left(\boldsymbol{D}\left(\boldsymbol{u}^2\right) + \mathrm{diag}\left(\boldsymbol{u}\right)\boldsymbol{D}\boldsymbol{u}\right) + \boldsymbol{M}^{-1}\boldsymbol{B}\boldsymbol{f}^* = 0, \qquad \boldsymbol{f}^* = \begin{bmatrix} \boldsymbol{f}_0^* \\ \vdots \\ \boldsymbol{f}_N^* \end{bmatrix}.$$

- Semi-discrete stability: multiply by $\boldsymbol{u}^T\boldsymbol{M}$, use $\boldsymbol{Q} = \boldsymbol{M}\boldsymbol{D}$ + diagonal matrices commute + SBP to eliminate volume terms

$$\boldsymbol{u}^T\boldsymbol{M}\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{dt}} + \frac{1}{3}\left(\boldsymbol{u}^T\boldsymbol{Q}\boldsymbol{u}^2 + \boldsymbol{u}^T\mathrm{diag}\left(\boldsymbol{u}\right)\boldsymbol{M}\boldsymbol{D}\boldsymbol{u}\right) + \boldsymbol{u}^T\boldsymbol{B}\boldsymbol{f}^* = 0.$$

Revisiting Burgers' equation: stable split formulations

- Rewrite Burgers' equation in split form

$$\frac{\partial u}{\partial t} + \frac{1}{3}\left(\frac{\partial u^2}{\partial x} + u\frac{\partial u}{\partial x}\right) = 0.$$

- SBP discretization of split formulation

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{dt}} + \frac{1}{3}\left(\boldsymbol{D}\left(\boldsymbol{u}^2\right) + \operatorname{diag}\left(\boldsymbol{u}\right)\boldsymbol{D}\boldsymbol{u}\right) + \boldsymbol{M}^{-1}\boldsymbol{B}\boldsymbol{f}^* = 0, \qquad \boldsymbol{f}^* = \begin{bmatrix} \boldsymbol{f}_0^* \\ \vdots \\ \boldsymbol{f}_N^* \end{bmatrix}.$$

- Semi-discrete stability: multiply by $\boldsymbol{u}^T\boldsymbol{M}$, use $\boldsymbol{Q} = \boldsymbol{M}\boldsymbol{D}$ + diagonal matrices commute + SBP to eliminate volume terms

$$\boldsymbol{u}^T\boldsymbol{M}\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{dt}} + \frac{1}{3}\left(\boldsymbol{u}^T\boldsymbol{Q}\boldsymbol{u}^2 + \left(\boldsymbol{u}^2\right)^T\boldsymbol{Q}\boldsymbol{u}\right) + \boldsymbol{u}^T\boldsymbol{B}\boldsymbol{f}^* = 0.$$

Revisiting Burgers' equation: stable split formulations

- Rewrite Burgers' equation in split form

$$\frac{\partial u}{\partial t} + \frac{1}{3}\left(\frac{\partial u^2}{\partial x} + u\frac{\partial u}{\partial x}\right) = 0.$$

- SBP discretization of split formulation

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{dt}} + \frac{1}{3}\left(\boldsymbol{D}\left(\boldsymbol{u}^2\right) + \mathrm{diag}\left(\boldsymbol{u}\right)\boldsymbol{D}\boldsymbol{u}\right) + \boldsymbol{M}^{-1}\boldsymbol{B}\boldsymbol{f}^* = 0, \qquad \boldsymbol{f}^* = \begin{bmatrix} \boldsymbol{f}_0^* \\ \vdots \\ \boldsymbol{f}_N^* \end{bmatrix}.$$

- Semi-discrete stability: multiply by $\boldsymbol{u}^T\boldsymbol{M}$, use $\boldsymbol{Q} = \boldsymbol{M}\boldsymbol{D}$ + diagonal matrices commute + SBP to eliminate volume terms

$$\boldsymbol{u}^T\boldsymbol{M}\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{dt}} + \frac{1}{3}\left(\boldsymbol{u}^T\boldsymbol{Q}\boldsymbol{u}^2 + \left(\boldsymbol{u}^2\right)^T\boldsymbol{Q}\boldsymbol{u}\right) + \boldsymbol{u}^T\boldsymbol{B}\boldsymbol{f}^* = 0.$$

Revisiting Burgers' equation: stable split formulations

- Rewrite Burgers' equation in split form

$$\frac{\partial u}{\partial t} + \frac{1}{3}\left(\frac{\partial u^2}{\partial x} + u\frac{\partial u}{\partial x}\right) = 0.$$

- SBP discretization of split formulation

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{dt}} + \frac{1}{3}\left(\boldsymbol{D}\left(\boldsymbol{u}^2\right) + \mathrm{diag}\left(\boldsymbol{u}\right)\boldsymbol{D}\boldsymbol{u}\right) + \boldsymbol{M}^{-1}\boldsymbol{B}\boldsymbol{f}^* = 0, \qquad \boldsymbol{f}^* = \begin{bmatrix} \boldsymbol{f}_0^* \\ \vdots \\ \boldsymbol{f}_N^* \end{bmatrix}.$$

- Semi-discrete stability: multiply by $\boldsymbol{u}^T\boldsymbol{M}$, use $\boldsymbol{Q} = \boldsymbol{M}\boldsymbol{D}$ + diagonal matrices commute + SBP to eliminate volume terms

$$\boldsymbol{u}^T\boldsymbol{M}\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{dt}} + \frac{1}{3}\left(\boldsymbol{u}^T\left(\boldsymbol{B} - \boldsymbol{Q}^T\right)\boldsymbol{u}^2 + \left(\boldsymbol{u}^2\right)^T\boldsymbol{Q}\boldsymbol{u}\right) + \boldsymbol{u}^T\boldsymbol{B}\boldsymbol{f}^* = 0.$$

Revisiting Burgers' equation: stable split formulations

- Rewrite Burgers' equation in split form

$$\frac{\partial u}{\partial t}+\frac{1}{3}\left(\frac{\partial u^2}{\partial x}+u\frac{\partial u}{\partial x}\right)=0.$$

- SBP discretization of split formulation

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{dt}}+\frac{1}{3}\left(\boldsymbol{D}\left(\boldsymbol{u}^2\right)+\operatorname{diag}\left(\boldsymbol{u}\right)\boldsymbol{D}\boldsymbol{u}\right)+\boldsymbol{M}^{-1}\boldsymbol{B}\boldsymbol{f}^*=0, \qquad \boldsymbol{f}^*=\begin{bmatrix}\boldsymbol{f}_0^*\\ \vdots\\ \boldsymbol{f}_N^*\end{bmatrix}.$$

- Semi-discrete stability: multiply by $\boldsymbol{u}^T\boldsymbol{M}$, use $\boldsymbol{Q}=\boldsymbol{M}\boldsymbol{D}$ + diagonal matrices commute + SBP to eliminate volume terms

$$\boldsymbol{u}^T\boldsymbol{M}\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{dt}}+\frac{1}{3}\left(\boldsymbol{u}^T\boldsymbol{B}\boldsymbol{u}^2-\boldsymbol{u}^T\boldsymbol{Q}^T\boldsymbol{u}^2+\left(\boldsymbol{u}^2\right)^T\boldsymbol{Q}\boldsymbol{u}\right)+\boldsymbol{u}^T\boldsymbol{B}\boldsymbol{f}^*=0.$$

Revisiting Burgers' equation: stable split formulations

- Rewrite Burgers' equation in split form

$$\frac{\partial u}{\partial t} + \frac{1}{3}\left(\frac{\partial u^2}{\partial x} + u\frac{\partial u}{\partial x}\right) = 0.$$

- SBP discretization of split formulation

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{dt}} + \frac{1}{3}\left(\boldsymbol{D}\left(\boldsymbol{u}^2\right) + \mathrm{diag}\left(\boldsymbol{u}\right)\boldsymbol{D}\boldsymbol{u}\right) + \boldsymbol{M}^{-1}\boldsymbol{B}\boldsymbol{f}^* = 0, \qquad \boldsymbol{f}^* = \begin{bmatrix} \boldsymbol{f}_0^* \\ \vdots \\ \boldsymbol{f}_N^* \end{bmatrix}.$$

- Semi-discrete stability: multiply by $\boldsymbol{u}^T\boldsymbol{M}$, use $\boldsymbol{Q} = \boldsymbol{M}\boldsymbol{D}$ + diagonal matrices commute + SBP to eliminate volume terms

$$\boldsymbol{u}^T\boldsymbol{M}\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{dt}} + \boldsymbol{u}^T\boldsymbol{B}\left(\frac{1}{3}\boldsymbol{u}^2 + \boldsymbol{f}^*\right) = 0.$$

Revisiting Burgers' equation: stable split formulations

- Rewrite Burgers' equation in split form

$$\frac{\partial u}{\partial t} + \frac{1}{3}\left(\frac{\partial u^2}{\partial x} + u\frac{\partial u}{\partial x}\right) = 0.$$

- SBP discretization of split formulation

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{dt}} + \frac{1}{3}\left(\boldsymbol{D}\left(\boldsymbol{u}^2\right) + \mathrm{diag}\left(\boldsymbol{u}\right)\boldsymbol{D}\boldsymbol{u}\right) + \boldsymbol{M}^{-1}\boldsymbol{B}\boldsymbol{f}^* = 0, \qquad \boldsymbol{f}^* = \begin{bmatrix} \boldsymbol{f}_0^* \\ \vdots \\ \boldsymbol{f}_N^* \end{bmatrix}.$$

- Semi-discrete stability: multiply by $\boldsymbol{u}^T\boldsymbol{M}$, use $\boldsymbol{Q} = \boldsymbol{M}\boldsymbol{D}$ + diagonal matrices commute + SBP to eliminate volume terms

$$\boldsymbol{u}^T\boldsymbol{M}\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{dt}} + \boldsymbol{u}^T\boldsymbol{B}\left(\frac{1}{3}\boldsymbol{u}^2 + \boldsymbol{f}^*\right) = 0.$$

Revisiting Burgers' equation: stable split formulations

- Rewrite Burgers' equation in split form

$$\frac{\partial u}{\partial t} + \frac{1}{3}\left(\frac{\partial u^2}{\partial x} + u\frac{\partial u}{\partial x}\right) = 0.$$

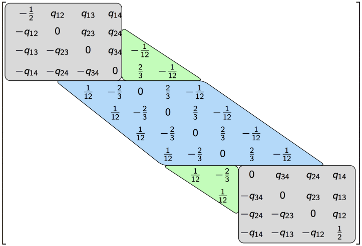
- SBP discretization of split formulation

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{dt}} + \frac{1}{3}\left(\boldsymbol{D}\left(\boldsymbol{u}^2\right) + \mathrm{diag}\left(\boldsymbol{u}\right)\boldsymbol{D}\boldsymbol{u}\right) + \boldsymbol{M}^{-1}\boldsymbol{B}\boldsymbol{f}^* = 0, \qquad \boldsymbol{f}^* = \begin{bmatrix} \boldsymbol{f}_0^* \\ \vdots \\ \boldsymbol{f}_N^* \end{bmatrix}.$$

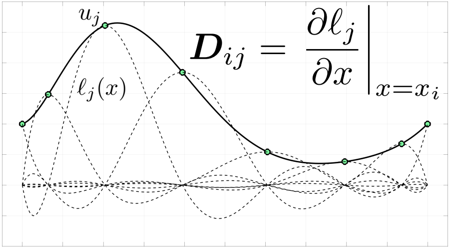
- Semi-discrete stability: multiply by $\boldsymbol{u}^T\boldsymbol{M}$, use $\boldsymbol{Q} = \boldsymbol{M}\boldsymbol{D}$ + diagonal matrices commute + SBP to eliminate volume terms

$$\frac{1}{2}\frac{\mathrm{d}}{\mathrm{dt}}\left(\boldsymbol{u}^T\boldsymbol{M}\boldsymbol{u}\right) = 0, \qquad \text{(for appropriate } \boldsymbol{f}^*\text{)}.$$

Higher order SBP approximations



(a) 1D matrix ($N = 2$, equispaced)



(b) 1D SBP ($N = 7$, GLL nodes)

- Solve for high order "diagonal-norm" SBP finite difference matrices.
- Nodal DG construction of diagonal-norm SBP matrices: Gauss-Legendre-Lobatto quadrature + nodal basis + mass lumping.

Figure courtesy of David C. Del Rey Fernandez.

Fisher and Carpenter (2013). *High-order ES finite difference schemes for nonlinear conservation laws: Finite domains.*

Gassner, Winters, and Kopriva (2016). *Split form nodal DG schemes with SBP property for the comp. Euler equations.*

Key idea behind entropy stable schemes

- Traditional (unstable) scheme, ignoring boundary conditions:

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{dt}} + \boldsymbol{D}\boldsymbol{f}(\boldsymbol{u}) = 0 \quad \Longrightarrow \quad \frac{\mathrm{d}\boldsymbol{u}_i}{\mathrm{dt}} + \sum_j \boldsymbol{D}_{ij}\boldsymbol{f}\left(\boldsymbol{u}_j\right) = 0.$$

- Flux differencing: $\boldsymbol{f}_S(\boldsymbol{u}_i, \boldsymbol{u}_j) = \frac{\boldsymbol{u}_i + \boldsymbol{u}_j}{2}$ recovers traditional scheme

$$\frac{\mathrm{d}\boldsymbol{u}_i}{\mathrm{dt}} + \sum_j \boldsymbol{D}_{ij} 2\boldsymbol{f}_S\left(\boldsymbol{u}_i, \boldsymbol{u}_j\right) = 0 \quad \Longrightarrow \quad \frac{\mathrm{d}\boldsymbol{u}}{\mathrm{dt}} + 2\left(\boldsymbol{D} \circ \boldsymbol{F}_S\right)\mathbf{1} = 0.$$

- Use "entropy conservative" finite volume numerical flux $\boldsymbol{f}_S(\boldsymbol{u}_L, \boldsymbol{u}_R)$.
- Semi-discrete entropy equality (add dissipation for entropy inequality)

$$\mathbf{1}^T \boldsymbol{M} \frac{\mathrm{d}S(\boldsymbol{u})}{\mathrm{dt}} + \mathbf{1}^T \boldsymbol{B}\left(\boldsymbol{v}^T \boldsymbol{f}(\boldsymbol{u}) - \psi(\boldsymbol{u})\right) = 0.$$

Key idea behind entropy stable schemes

- Traditional (unstable) scheme, ignoring boundary conditions:

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{dt}} + \boldsymbol{D}\boldsymbol{f}(\boldsymbol{u}) = 0 \quad \Longrightarrow \quad \frac{\mathrm{d}\boldsymbol{u}_i}{\mathrm{dt}} + \sum_j \boldsymbol{D}_{ij}\boldsymbol{f}\left(\boldsymbol{u}_j\right) = 0.$$

- Flux differencing: $\boldsymbol{f}_S(\boldsymbol{u}_i, \boldsymbol{u}_j) = \frac{\boldsymbol{u}_i + \boldsymbol{u}_j}{2}$ recovers traditional scheme

$$\frac{\mathrm{d}\boldsymbol{u}_i}{\mathrm{dt}} + \sum_j \boldsymbol{D}_{ij} 2\boldsymbol{f}_S\left(\boldsymbol{u}_i, \boldsymbol{u}_j\right) = 0 \quad \Longrightarrow \quad \frac{\mathrm{d}\boldsymbol{u}}{\mathrm{dt}} + 2\left(\boldsymbol{D} \circ \boldsymbol{F}_S\right)\boldsymbol{1} = 0.$$

- Use "entropy conservative" finite volume numerical flux $\boldsymbol{f}_S(\boldsymbol{u}_L, \boldsymbol{u}_R)$.
- Semi-discrete entropy equality (add dissipation for entropy inequality)

$$\boldsymbol{1}^T \boldsymbol{M} \frac{\mathrm{d}S(\boldsymbol{u})}{\mathrm{dt}} + \boldsymbol{1}^T \boldsymbol{B}\left(\boldsymbol{v}^T \boldsymbol{f}(\boldsymbol{u}) - \psi(\boldsymbol{u})\right) = 0.$$

Key idea behind entropy stable schemes

- Traditional (unstable) scheme, ignoring boundary conditions:

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{dt}} + \boldsymbol{D}\boldsymbol{f}(\boldsymbol{u}) = 0 \quad \Longrightarrow \quad \frac{\mathrm{d}\boldsymbol{u}_i}{\mathrm{dt}} + \sum_j \boldsymbol{D}_{ij}\boldsymbol{f}\left(\boldsymbol{u}_j\right) = 0.$$

- Flux differencing: $\boldsymbol{f}_S(\boldsymbol{u}_i, \boldsymbol{u}_j) = \frac{\boldsymbol{u}_i + \boldsymbol{u}_j}{2}$ recovers traditional scheme

$$\frac{\mathrm{d}\boldsymbol{u}_i}{\mathrm{dt}} + \sum_j \boldsymbol{D}_{ij} 2\boldsymbol{f}_S\left(\boldsymbol{u}_i, \boldsymbol{u}_j\right) = 0 \quad \Longrightarrow \quad \frac{\mathrm{d}\boldsymbol{u}}{\mathrm{dt}} + 2\left(\boldsymbol{D} \circ \boldsymbol{F}_S\right)\boldsymbol{1} = 0.$$

- Use “entropy conservative” finite volume numerical flux $\boldsymbol{f}_S(\boldsymbol{u}_L, \boldsymbol{u}_R)$.

- Semi-discrete entropy equality (add dissipation for entropy inequality)

$$\boldsymbol{1}^T\boldsymbol{M}\frac{\mathrm{d}S(\boldsymbol{u})}{\mathrm{dt}} + \boldsymbol{1}^T\boldsymbol{B}\left(\boldsymbol{v}^T\boldsymbol{f}(\boldsymbol{u}) - \psi(\boldsymbol{u})\right) = 0.$$

Key idea behind entropy stable schemes

- Traditional (unstable) scheme, ignoring boundary conditions:

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{dt}} + \boldsymbol{D}\boldsymbol{f}(\boldsymbol{u}) = 0 \quad \Longrightarrow \quad \frac{\mathrm{d}\boldsymbol{u}_i}{\mathrm{dt}} + \sum_j \boldsymbol{D}_{ij}\boldsymbol{f}\left(\boldsymbol{u}_j\right) = 0.$$

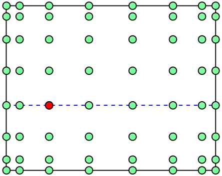
- Flux differencing: $\boldsymbol{f}_S(\boldsymbol{u}_i, \boldsymbol{u}_j) = \frac{\boldsymbol{u}_i + \boldsymbol{u}_j}{2}$ recovers traditional scheme

$$\frac{\mathrm{d}\boldsymbol{u}_i}{\mathrm{dt}} + \sum_j \boldsymbol{D}_{ij} 2\boldsymbol{f}_S\left(\boldsymbol{u}_i, \boldsymbol{u}_j\right) = 0 \quad \Longrightarrow \quad \frac{\mathrm{d}\boldsymbol{u}}{\mathrm{dt}} + 2\left(\boldsymbol{D} \circ \boldsymbol{F}_S\right)\boldsymbol{1} = 0.$$

- Use "entropy conservative" finite volume numerical flux $\boldsymbol{f}_S(\boldsymbol{u}_L, \boldsymbol{u}_R)$.
- Semi-discrete entropy equality (add dissipation for entropy inequality)

$$\boldsymbol{1}^T \boldsymbol{M} \frac{\mathrm{d}S(\boldsymbol{u})}{\mathrm{dt}} + \boldsymbol{1}^T \boldsymbol{B}\left(\boldsymbol{v}^T \boldsymbol{f}(\boldsymbol{u}) - \psi(\boldsymbol{u})\right) = 0.$$

Entropy stable SBP discretizations: current/challenges

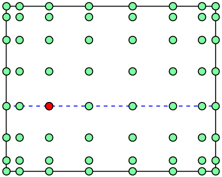


(a) GLL collocation

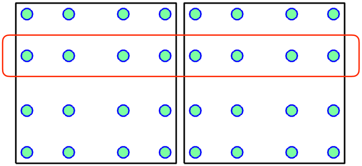
- (Current) Discrete entropy inequality using high order GLL hexes.
- Gauss quadrature: more accurate but expensive coupling conditions.
- Tetrahedra, wedges, pyramids? Over-integration?

Gassner, Winters, and Kopriva (2016). *Split form nodal DG schemes with SBP property for the comp. Euler equations.*
Carpenter et al. (2014). *Entropy stable spectral collocation schemes for the NS equations: discontinuous interfaces.*

Entropy stable SBP discretizations: current/challenges



(a) GLL collocation

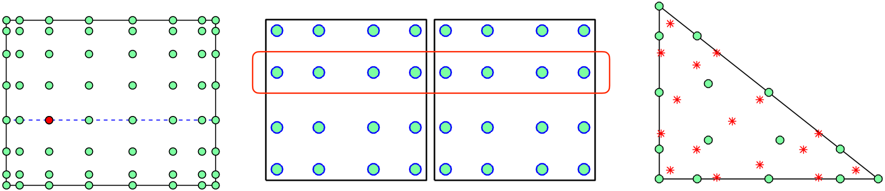


(b) Gauss nodes element coupling

- (Current) Discrete entropy inequality using high order GLL hexes.
- Gauss quadrature: more accurate but expensive coupling conditions.
- Tetrahedra, wedges, pyramids? Over-integration?

Gassner, Winters, and Kopriva (2016). *Split form nodal DG schemes with SBP property for the comp. Euler equations.*
Carpenter et al. (2014). *Entropy stable spectral collocation schemes for the NS equations: discontinuous interfaces.*

Entropy stable SBP discretizations: current/challenges

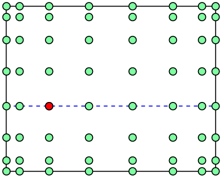


(a) GLL collocation (b) Gauss nodes element coupling (c) Nodes vs quadrature

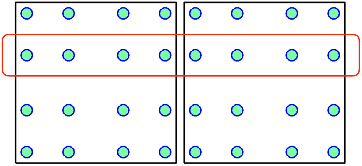
- (Current) Discrete entropy inequality using high order GLL hexes.
- Gauss quadrature: more accurate but expensive coupling conditions.
- Tetrahedra, wedges, pyramids? Over-integration?

Gassner, Winters, and Kopriva (2016). *Split form nodal DG schemes with SBP property for the comp. Euler equations.*
Carpenter et al. (2014). *Entropy stable spectral collocation schemes for the NS equations: discontinuous interfaces.*

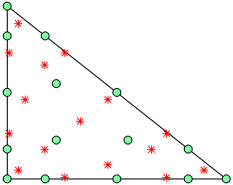
Entropy stable SBP discretizations: current/challenges



(a) GLL collocation



(b) Gauss nodes element coupling



(c) Nodes vs quadrature

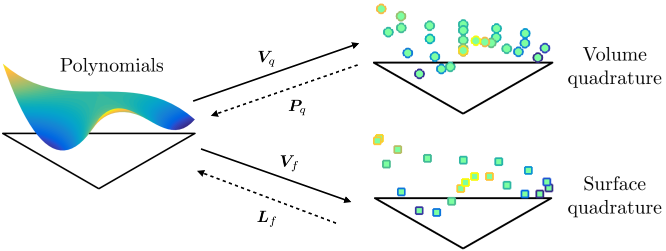
- (Current) Discrete entropy inequality using high order GLL hexes.
- Gauss quadrature: more accurate but expensive coupling conditions.
- Tetrahedra, wedges, pyramids? Over-integration?

Goal: entropy stable high order DG with compact stencils using arbitrary basis functions and volume/surface quadrature points.

Gassner, Winters, and Kopriva (2016). *Split form nodal DG schemes with SBP property for the comp. Euler equations.*

Carpenter et al. (2014). *Entropy stable spectral collocation schemes for the NS equations: discontinuous interfaces.*

Quadrature-based matrices for polynomial bases



- Assume degree $2N$ volume, surface quadratures $(\boldsymbol{x}_i^q, \boldsymbol{w}_i^q)$, $(\boldsymbol{x}_i^f, \boldsymbol{w}_i^f)$, and basis $\phi_1, \ldots, \phi_{N_p}$. Define interpolation matrices $\boldsymbol{V}_q, \boldsymbol{V}_f$

$$(\boldsymbol{V}_q)_{ij} = \phi_j(\boldsymbol{x}_i^q), \qquad (\boldsymbol{V}_f)_{ij} = \phi_j(\boldsymbol{x}_i^f).$$

- Introduce quadrature-based L^2 projection and lifting matrices

$$\boldsymbol{P}_q = \boldsymbol{M}^{-1}\boldsymbol{V}_q^T\boldsymbol{W}, \qquad \boldsymbol{L}_f = \boldsymbol{M}^{-1}\boldsymbol{V}_f^T\boldsymbol{W}_f,$$
$$\boldsymbol{W} = \operatorname{diag}(\boldsymbol{w}^q), \qquad \boldsymbol{W}_f = \operatorname{diag}\left(\boldsymbol{w}^f\right).$$

Quadrature-based differentiation matrices

- Matrix $\boldsymbol{D}_q^i$: evaluates derivative of L^2 projection at points $\boldsymbol{x}^q$.

$$\boldsymbol{D}_q^i = \boldsymbol{V}_q \boldsymbol{D}^i \boldsymbol{P}_q, \qquad \boldsymbol{D}^i \quad \text{exactly differentiates polynomials.}$$

- Summation-by-parts involving L^2 projection:

$$\boldsymbol{W}\boldsymbol{D}_q^i + \left(\boldsymbol{W}\boldsymbol{D}_q^i\right)^T = \left(\boldsymbol{V}_f \boldsymbol{P}_q\right)^T \boldsymbol{W}_f \mathrm{diag}\left(\boldsymbol{n}_i\right) \boldsymbol{V}_f \boldsymbol{P}_q.$$

- Equivalent to integration-by-parts + quadrature: for $u, v \in L^2\left(\widehat{D}\right)$

$$\int_{\widehat{D}} \frac{\partial P_N u}{\partial x_i} v + \int_{\widehat{D}} u \frac{\partial P_N v}{\partial x_i} = \int_{\partial \widehat{D}} \left(P_N u\right)\left(P_N v\right) \widehat{n}_i$$

- Quadrature may not contain boundary points: complicated interface terms for coupling between neighboring elements or imposing BCs.

A "decoupled" block SBP operator

- Approx. derivatives also using boundary traces (compact coupling).
- On an element D^k with unit normal vector $\boldsymbol{n}$: approximate derivative with respect to the ith coordinate.

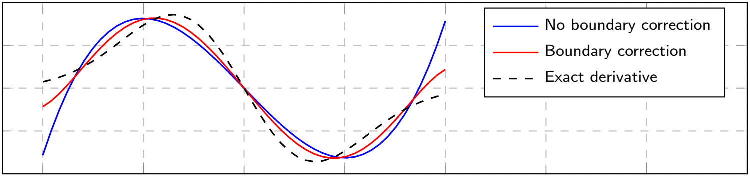
$$\boldsymbol{D}_N^i = \begin{bmatrix} \boldsymbol{D}_q^i - \frac{1}{2}\boldsymbol{V}_q \boldsymbol{L}_f \operatorname{diag}(\boldsymbol{n}_i) \boldsymbol{V}_f \boldsymbol{P}_q & \frac{1}{2}\boldsymbol{V}_q \boldsymbol{L}_f \operatorname{diag}(\boldsymbol{n}_i) \\ -\frac{1}{2}\operatorname{diag}(\boldsymbol{n}_i) \boldsymbol{V}_f \boldsymbol{P}_q & \frac{1}{2}\operatorname{diag}(\boldsymbol{n}_i) \end{bmatrix},$$

- $\boldsymbol{D}_N^i$ satisfies a summation-by-parts (SBP) property

$$\boldsymbol{Q}_N^i = \begin{bmatrix} \boldsymbol{W} & \\ & \boldsymbol{W}_f \end{bmatrix} \boldsymbol{D}_N^i, \qquad \boldsymbol{B}_N = \begin{bmatrix} 0 & \\ & \boldsymbol{W}_f \boldsymbol{n}_i \end{bmatrix},$$

$$\boxed{\boldsymbol{Q}_N^i + \left(\boldsymbol{Q}_N^i\right)^T = \boldsymbol{B}_N} \sim \boxed{\int_{D^k} \frac{\partial f}{\partial x_i} g + f \frac{\partial g}{\partial x_i} = \int_{\partial D^k} fg \boldsymbol{n}_i}.$$

Decoupled SBP operators: adding boundary corrections



- $\boldsymbol{D}_N^i$ produces a high order approximation of $f\frac{\partial g}{\partial x}$ at $\boldsymbol{x} = [\boldsymbol{x}^q, \boldsymbol{x}^f]$.

$$f\frac{\partial g}{\partial x} \approx \begin{bmatrix} \boldsymbol{P}_q & \boldsymbol{L}_f \end{bmatrix} \operatorname{diag}(\boldsymbol{f})\, \boldsymbol{D}_N \boldsymbol{g}, \qquad \boldsymbol{f}_i, \boldsymbol{g}_i = f(\boldsymbol{x}_i), g(\boldsymbol{x}_i).$$

- Equivalent to a skew-symmetric variational problem for $u(\boldsymbol{x}) \approx f\frac{\partial g}{\partial x}$ (P_N is the degree N polynomial L^2 projection).

$$\int_{D^k} u(\boldsymbol{x})v(\boldsymbol{x}) = \int_{D^k} f\frac{\partial P_N g}{\partial x} v + \int_{\partial D^k} (g - P_N g)\frac{(fv + P_N(fv))}{2}.$$

Talk outline

1 Stability of high order DG: linear vs nonlinear PDEs

2 Summation by parts finite differences and high order DG

3 Entropy stable formulations and flux differencing

4 Numerical experiments
- 1D experiments
- Triangular and tetrahedral meshes
- Quadrilateral and hexahedral meshes

Burgers' equation: energy stable formulations

- Revisit split form of Burgers' equation:

$$\frac{\partial u}{\partial t} + \frac{1}{3}\left(\frac{\partial u^2}{\partial x} + u\frac{\partial u}{\partial x}\right) = 0$$

- Stable DG method: let $u(x) = \sum_j \widehat{\boldsymbol{u}}_j \phi(x)$. Find $\widehat{\boldsymbol{u}}$ such that

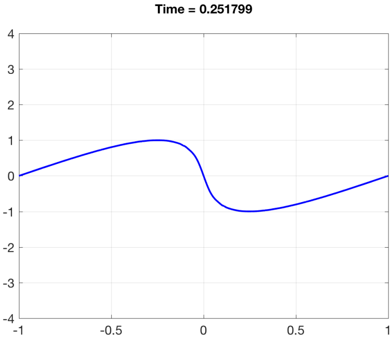
$$\boldsymbol{u} = \begin{bmatrix} \boldsymbol{V}_q \\ \boldsymbol{V}_f \end{bmatrix} \widehat{\boldsymbol{u}}, \qquad \boldsymbol{f}^* = \boldsymbol{f}^*(u^+, u) = \text{numerical flux}$$

$$\frac{\mathrm{d}\widehat{\boldsymbol{u}}}{\mathrm{dt}} + \frac{1}{3}\begin{bmatrix} \boldsymbol{P}_q & \boldsymbol{L}_f \end{bmatrix}\left(\boldsymbol{D}_N\left(\boldsymbol{u}^2\right) + \operatorname{diag}\left(\boldsymbol{u}\right)\boldsymbol{D}_N\boldsymbol{u}\right) + \boldsymbol{L}_f(\boldsymbol{f}^*) = 0.$$

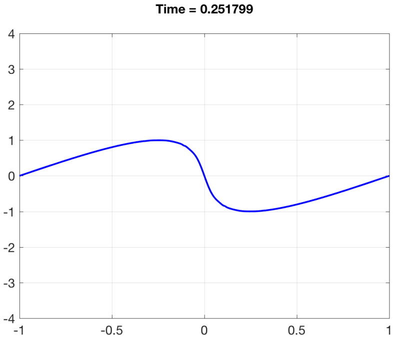
- Energy stability: multiply by $\widehat{\boldsymbol{u}}^T \boldsymbol{M}$, use SBP, sum over D^k

$$\sum_k \frac{1}{2}\frac{\mathrm{d}}{\mathrm{dt}}\widehat{\boldsymbol{u}}^T\boldsymbol{M}\widehat{\boldsymbol{u}} = \sum_k \frac{1}{2}\frac{\partial}{\partial t}\left\|u\right\|^2_{L^2(D^k)} \leq 0.$$

Burgers' equation: energy stable shock solution

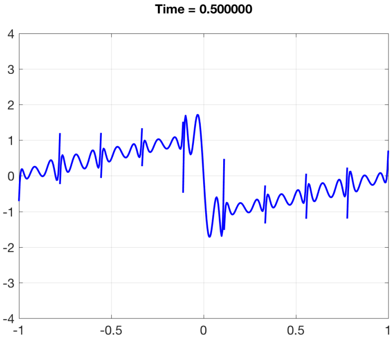


(a) $\tau = 0$

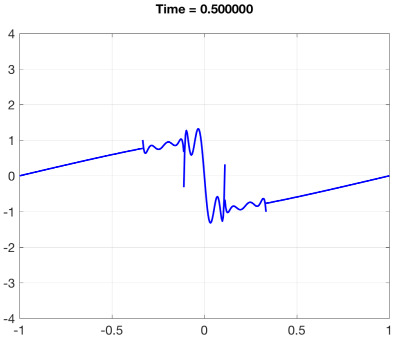


(b) $\tau = 1$

Burgers' equation: energy stable shock solution

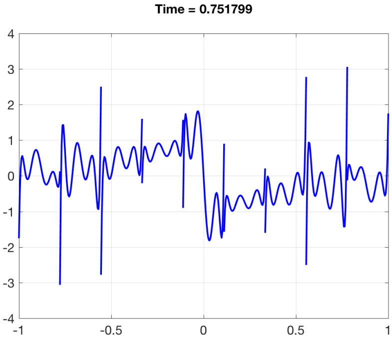


(a) $\tau = 0$

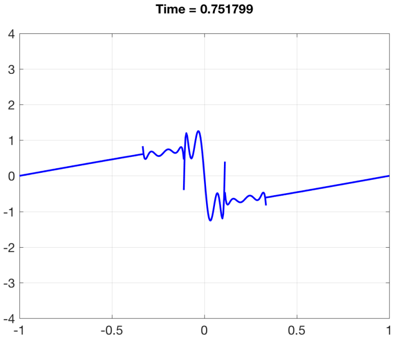


(b) $\tau = 1$

Burgers' equation: energy stable shock solution

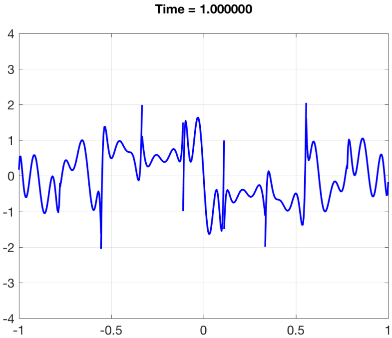


(a) $\tau = 0$

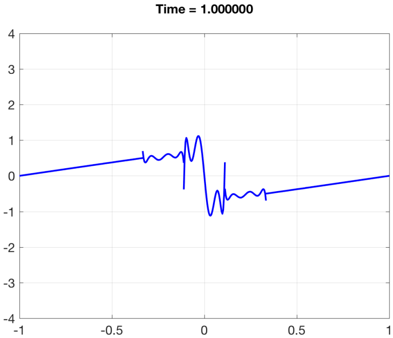


(b) $\tau = 1$

Burgers' equation: energy stable shock solution

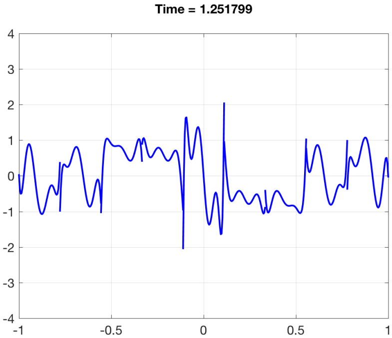


(a) $\tau = 0$

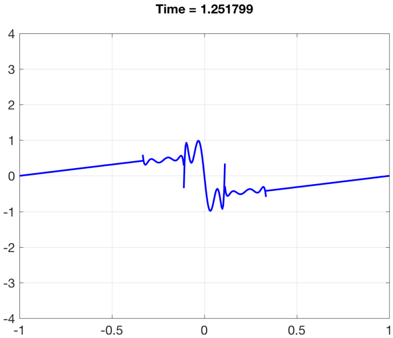


(b) $\tau = 1$

Burgers' equation: energy stable shock solution

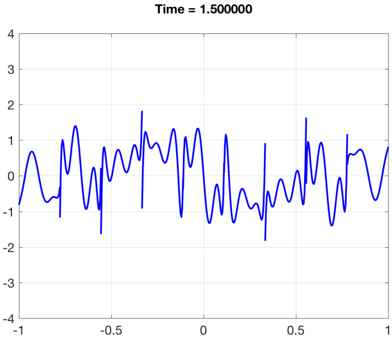


(a) $\tau = 0$

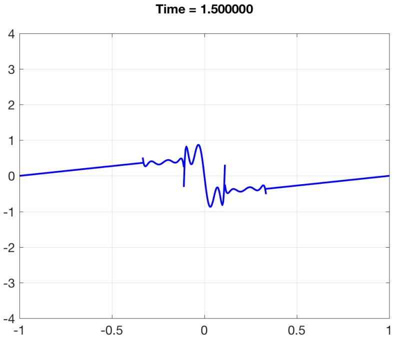


(b) $\tau = 1$

Burgers' equation: energy stable shock solution

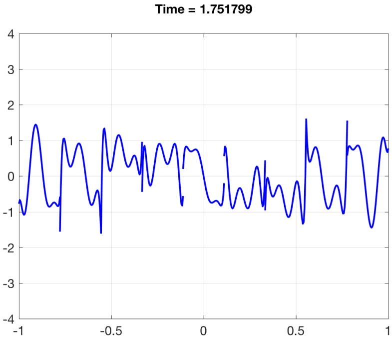


(a) $\tau = 0$

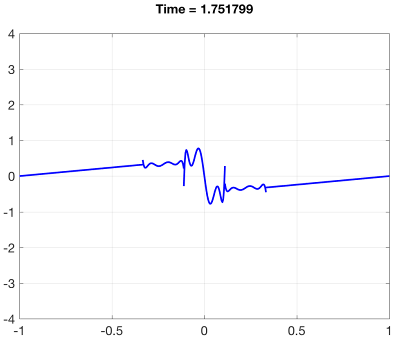


(b) $\tau = 1$

Burgers' equation: energy stable shock solution

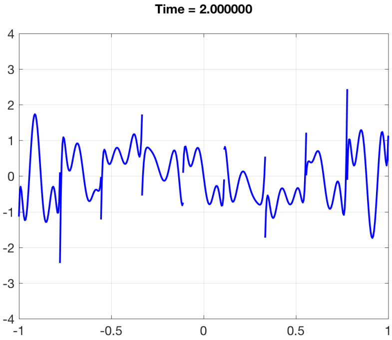


(a) $\tau = 0$

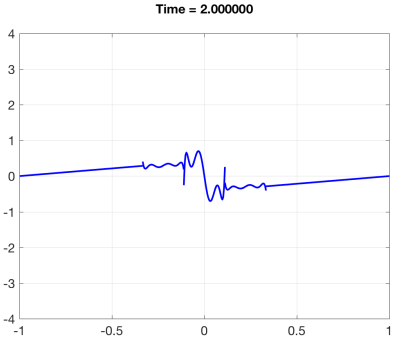


(b) $\tau = 1$

Burgers' equation: energy stable shock solution

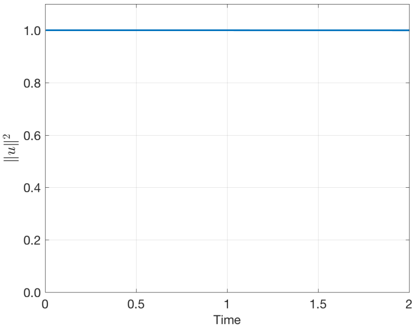


(a) $\tau = 0$

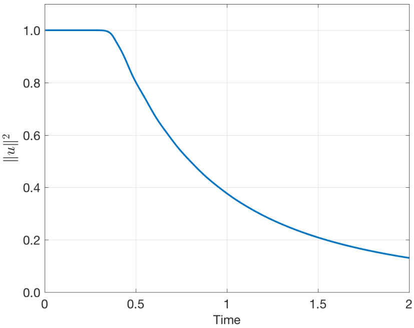


(b) $\tau = 1$

Burgers' equation: energy stable shock solution



(a) Energy conservative ($\tau = 0$)



(b) Energy stable ($\tau = 1$)

Flux differencing: entropy conservative finite volume fluxes

- Tadmor's entropy conservative (mean value) numerical flux

$$
\begin{aligned}
\boldsymbol{f}_S(\boldsymbol{u},\boldsymbol{u}) &= \boldsymbol{f}(\boldsymbol{u}), && \text{(consistency)}\\
\boldsymbol{f}_S(\boldsymbol{u},\boldsymbol{v}) &= \boldsymbol{f}_S(\boldsymbol{v},\boldsymbol{u}), && \text{(symmetry)}\\
(\boldsymbol{v}_L-\boldsymbol{v}_R)^T\,\boldsymbol{f}\,(\boldsymbol{u}_L,\boldsymbol{u}_R) &= \psi_L-\psi_R, && \text{(conservation)}.
\end{aligned}
$$

- Example: entropy conservative flux for Burgers' equation

$$
f_S(u_L,u_R)=\frac{1}{6}\left(u_L^2+u_Lu_R+u_R^2\right).
$$

- Flux differencing: using finite volume numerical fluxes to evaluate high order derivatives in DG methods.

Tadmor, Eitan (1987). *The numerical viscosity of entropy stable schemes for systems of conservation laws. I.*

Flux differencing: entropy conservative finite volume fluxes

- Tadmor's entropy conservative (mean value) numerical flux

$$\begin{aligned} \boldsymbol{f}_S(\boldsymbol{u}, \boldsymbol{u}) &= \boldsymbol{f}(\boldsymbol{u}), && \text{(consistency)} \\ \boldsymbol{f}_S(\boldsymbol{u}, \boldsymbol{v}) &= \boldsymbol{f}_S(\boldsymbol{v}, \boldsymbol{u}), && \text{(symmetry)} \\ (\boldsymbol{v}_L - \boldsymbol{v}_R)^T \boldsymbol{f}(\boldsymbol{u}_L, \boldsymbol{u}_R) &= \psi_L - \psi_R, && \text{(conservation).} \end{aligned}$$

- Example: entropy conservative flux for Burgers' equation

$$f_S(u_L, u_R) = \frac{1}{6}\left(u_L^2 + u_L u_R + u_R^2\right).$$

- Flux differencing: using finite volume numerical fluxes to evaluate high order derivatives in DG methods.

Tadmor, Eitan (1987). *The numerical viscosity of entropy stable schemes for systems of conservation laws. I.*

Flux differencing: entropy conservative finite volume fluxes

- Tadmor's entropy conservative (mean value) numerical flux

$$\begin{aligned} \boldsymbol{f}_S(\boldsymbol{u},\boldsymbol{u}) &= \boldsymbol{f}(\boldsymbol{u}), && \text{(consistency)} \\ \boldsymbol{f}_S(\boldsymbol{u},\boldsymbol{v}) &= \boldsymbol{f}_S(\boldsymbol{v},\boldsymbol{u}), && \text{(symmetry)} \\ (\boldsymbol{v}_L - \boldsymbol{v}_R)^T \boldsymbol{f}(\boldsymbol{u}_L,\boldsymbol{u}_R) &= \psi_L - \psi_R, && \text{(conservation).} \end{aligned}$$

- Example: entropy conservative flux for Burgers' equation

$$f_S(u_L,u_R) = \frac{1}{6}\left(u_L^2 + u_L u_R + u_R^2\right).$$

- Flux differencing: using finite volume numerical fluxes to evaluate high order derivatives in DG methods.

Tadmor, Eitan (1987). *The numerical viscosity of entropy stable schemes for systems of conservation laws. I.*

Flux differencing: recovering split formulations

- Entropy conservative flux for Burgers' equation

$$f_S(u_L, u_R) = \frac{1}{6}\left(u_L^2 + u_L u_R + u_R^2\right).$$

- Flux differencing: let $u_L = u(x), u_R = u(y)$

$$\frac{\partial f(u)}{\partial x} \Longrightarrow 2\frac{\partial f_S\left(u(x), u(y)\right)}{\partial x}\bigg|_{y=x}$$

- Recovering the Burgers' split formulation

$$f_S(u(x), u(y)) = \frac{1}{6}\left(u(x)^2 + u(x)u(y) + u(y)^2\right)$$

$$2\frac{\partial f_S\left(u(x), u(y)\right)}{\partial x}\bigg|_{y=x} = \frac{1}{3}\frac{\partial u^2}{\partial x} + \frac{1}{3}u\frac{\partial u}{\partial x} + \frac{1}{3}u^2\frac{\partial \cancel{1}}{\cancel{\partial x}}.$$

Flux differencing: recovering split formulations

- Entropy conservative flux for Burgers' equation

$$f_S(u_L, u_R) = \frac{1}{6}\left(u_L^2 + u_L u_R + u_R^2\right).$$

- Flux differencing: let $u_L = u(x), u_R = u(y)$

$$\frac{\partial f(u)}{\partial x} \Longrightarrow 2\frac{\partial f_S\left(u(x), u(y)\right)}{\partial x}\bigg|_{y=x}$$

- Recovering the Burgers' split formulation

$$f_S(u(x), u(y)) = \frac{1}{6}\left(u(x)^2 + u(x)u(y) + u(y)^2\right)$$

$$2\frac{\partial f_S\left(u(x), u(y)\right)}{\partial x}\bigg|_{y=x} = \frac{1}{3}\frac{\partial u^2}{\partial x} + \frac{1}{3}u\frac{\partial u}{\partial x} + \frac{1}{3}u^2\cancel{\frac{\partial 1}{\partial x}}.$$

Flux differencing: beyond split formulations

- Fluxes do not necessarily correspond to split formulations!
- Example: entropy conservative flux for 1D compressible Euler

$$
\begin{aligned}
f_S^1(\boldsymbol{u}_L, \boldsymbol{u}_R) &= \{\{\rho\}\}^{\log} \{\{u\}\} \\
f_S^2(\boldsymbol{u}_L, \boldsymbol{u}_R) &= \frac{\{\{\rho\}\}}{2\{\{\beta\}\}} + \{\{u\}\} f_S^1 \\
f_S^3(\boldsymbol{u}_L, \boldsymbol{u}_R) &= f_S^1 \left(\frac{1}{2(\gamma-1)\{\{\beta\}\}^{\log}} - \frac{1}{2}\{\{u^2\}\} \right) + \{\{u\}\} f_S^2,
\end{aligned}
$$

- Rational functions: logarithmic mean and "inverse temperature" β

$$
\{\{u\}\}^{\log} = \frac{u_L - u_R}{\log u_L - \log u_R}, \qquad \beta = \frac{\rho}{2p}.
$$

Chandreshekar (2013), *Kinetic energy preserving and entropy stable FV schemes for comp. Euler and NS equations.*

Flux differencing: implementational details

- Define $\boldsymbol{F}_S$ by evaluating $\boldsymbol{f}_S$ at all combinations of quadrature points

$$(\boldsymbol{F}_S)_{ij} = \boldsymbol{f}_S\left(u(\boldsymbol{x}_i), u(\boldsymbol{x}_j)\right), \qquad \boldsymbol{x} = \left[\boldsymbol{x}^q, \boldsymbol{x}^f\right]^T.$$

- Replace $\frac{\partial}{\partial x}$ with $\boldsymbol{D}_N$ + projection and lifting matrices.

$$2\frac{\partial f_S\left(u(x), u(y)\right)}{\partial x}\bigg|_{y=x} \Longrightarrow \begin{bmatrix} \boldsymbol{P}_q & \boldsymbol{L}_f \end{bmatrix} \operatorname{diag}(2\boldsymbol{D}_N \boldsymbol{F}_S).$$

- Efficient Hadamard product reformulation of flux differencing (efficient on-the-fly evaluation of $\boldsymbol{F}_S$)

$$\operatorname{diag}(2\boldsymbol{D}_N \boldsymbol{F}_S) = \left(2\boldsymbol{D}_N \circ \boldsymbol{F}_S\right)\mathbf{1}.$$

Flux differencing: avoiding the chain rule

- Test with entropy variables $\widetilde{\boldsymbol{v}}$, integrate, and use SBP property:

$$\widetilde{\boldsymbol{v}}^T \left(2\boldsymbol{Q}_N \circ \boldsymbol{F}_S\right)\boldsymbol{1} = \widetilde{\boldsymbol{v}}^T \left(\left(\begin{bmatrix} 0 & \\ & \boldsymbol{W}_f \boldsymbol{n} \end{bmatrix} + \boldsymbol{Q}_N - \boldsymbol{Q}_N^T\right) \circ \boldsymbol{F}_S\right)\boldsymbol{1}.$$

- Only boundary terms appear in final estimate; volume terms become boundary terms using properties of $\left(\boldsymbol{F}_S\right)_{ij} = \boldsymbol{f}_S\left(\widetilde{\boldsymbol{u}}_i, \widetilde{\boldsymbol{u}}_j\right)$

$$\begin{aligned}\widetilde{\boldsymbol{v}}^T \left(\left(\boldsymbol{Q}_N - \boldsymbol{Q}_N^T\right) \circ \boldsymbol{F}_S\right)\boldsymbol{1} &= \widetilde{\boldsymbol{v}}^T \left(\boldsymbol{Q}_N \circ \boldsymbol{F}_S\right)\boldsymbol{1} - \boldsymbol{1}^T \left(\boldsymbol{Q}_N \circ \boldsymbol{F}_S\right)\widetilde{\boldsymbol{v}} \\ &= \sum_{i,j} \left(\boldsymbol{Q}_N\right)_{ij} \left(\widetilde{\boldsymbol{v}}_i - \widetilde{\boldsymbol{v}}_j\right)^T \boldsymbol{f}_S\left(\widetilde{\boldsymbol{u}}_i, \widetilde{\boldsymbol{u}}_j\right).\end{aligned}$$

- Proof requires $\widetilde{\boldsymbol{v}} = \boldsymbol{v}(\widetilde{\boldsymbol{u}})$; the entropy variables $\widetilde{\boldsymbol{v}}$ must be a function of the conservative variables $\widetilde{\boldsymbol{u}}$.

Flux differencing: avoiding the chain rule

- Test with entropy variables $\widetilde{\boldsymbol{v}}$, integrate, and use SBP property:

$$\widetilde{\boldsymbol{v}}^T\left(2\boldsymbol{Q}_N \circ \boldsymbol{F}_S\right)\boldsymbol{1} = \widetilde{\boldsymbol{v}}^T\left(\left(\begin{bmatrix} 0 & \\ & \boldsymbol{W}_f\boldsymbol{n} \end{bmatrix} + \boldsymbol{Q}_N - \boldsymbol{Q}_N^T\right)\circ \boldsymbol{F}_S\right)\boldsymbol{1}.$$

- Only boundary terms appear in final estimate; volume terms become boundary terms using properties of $\left(\boldsymbol{F}_S\right)_{ij} = \boldsymbol{f}_S\left(\widetilde{\boldsymbol{u}}_i, \widetilde{\boldsymbol{u}}_j\right)$

$$\begin{aligned}\widetilde{\boldsymbol{v}}^T\left(\left(\boldsymbol{Q}_N - \boldsymbol{Q}_N^T\right)\circ \boldsymbol{F}_S\right)\boldsymbol{1} &= \widetilde{\boldsymbol{v}}^T\left(\boldsymbol{Q}_N \circ \boldsymbol{F}_S\right)\boldsymbol{1} - \boldsymbol{1}^T\left(\boldsymbol{Q}_N \circ \boldsymbol{F}_S\right)\widetilde{\boldsymbol{v}} \\ &= \sum_{i,j}\left(\boldsymbol{Q}_N\right)_{ij}\left(\psi(\widetilde{\boldsymbol{u}}_i) - \psi(\widetilde{\boldsymbol{u}}_j)\right).\end{aligned}$$

- Proof requires $\widetilde{\boldsymbol{v}} = \boldsymbol{v}(\widetilde{\boldsymbol{u}})$; the entropy variables $\widetilde{\boldsymbol{v}}$ must be a function of the conservative variables $\widetilde{\boldsymbol{u}}$.

Flux differencing: avoiding the chain rule

- Test with entropy variables $\widetilde{\boldsymbol{v}}$, integrate, and use SBP property:

$$\widetilde{\boldsymbol{v}}^T \left(2\boldsymbol{Q}_N \circ \boldsymbol{F}_S\right)\mathbf{1} = \widetilde{\boldsymbol{v}}^T \left(\left(\begin{bmatrix} 0 & \\ & \boldsymbol{W}_f \boldsymbol{n} \end{bmatrix} + \boldsymbol{Q}_N - \boldsymbol{Q}_N^T\right) \circ \boldsymbol{F}_S\right)\mathbf{1}.$$

- Only boundary terms appear in final estimate; volume terms become boundary terms using properties of $\left(\boldsymbol{F}_S\right)_{ij} = \boldsymbol{f}_S\left(\widetilde{\boldsymbol{u}}_i, \widetilde{\boldsymbol{u}}_j\right)$

$$\begin{aligned} \widetilde{\boldsymbol{v}}^T \left(\left(\boldsymbol{Q}_N - \boldsymbol{Q}_N^T\right) \circ \boldsymbol{F}_S\right)\mathbf{1} &= \widetilde{\boldsymbol{v}}^T \left(\boldsymbol{Q}_N \circ \boldsymbol{F}_S\right)\mathbf{1} - \mathbf{1}^T \left(\boldsymbol{Q}_N \circ \boldsymbol{F}_S\right)\widetilde{\boldsymbol{v}} \\ &= \mathbf{1}^T \boldsymbol{Q}_N \psi - \psi^T \boldsymbol{Q}_N \mathbf{1} = \mathbf{1}^T \boldsymbol{Q}_N \psi \end{aligned}$$

- Proof requires $\widetilde{\boldsymbol{v}} = \boldsymbol{v}(\widetilde{\boldsymbol{u}})$; the entropy variables $\widetilde{\boldsymbol{v}}$ must be a function of the conservative variables $\widetilde{\boldsymbol{u}}$.

Flux differencing: avoiding the chain rule

- Test with entropy variables $\widetilde{\boldsymbol{v}}$, integrate, and use SBP property:

$$\widetilde{\boldsymbol{v}}^T \left(2\boldsymbol{Q}_N \circ \boldsymbol{F}_S\right)\mathbf{1} = \widetilde{\boldsymbol{v}}^T \left(\left(\begin{bmatrix} 0 & \\ & \boldsymbol{W}_f \boldsymbol{n} \end{bmatrix} + \boldsymbol{Q}_N - \boldsymbol{Q}_N^T\right) \circ \boldsymbol{F}_S\right)\mathbf{1}.$$

- Only boundary terms appear in final estimate; volume terms become boundary terms using properties of $\left(\boldsymbol{F}_S\right)_{ij} = \boldsymbol{f}_S\left(\widetilde{\boldsymbol{u}}_i, \widetilde{\boldsymbol{u}}_j\right)$

$$\begin{aligned}\widetilde{\boldsymbol{v}}^T \left(\left(\boldsymbol{Q}_N - \boldsymbol{Q}_N^T\right) \circ \boldsymbol{F}_S\right)\mathbf{1} &= \widetilde{\boldsymbol{v}}^T \left(\boldsymbol{Q}_N \circ \boldsymbol{F}_S\right)\mathbf{1} - \mathbf{1}^T \left(\boldsymbol{Q}_N \circ \boldsymbol{F}_S\right)\widetilde{\boldsymbol{v}} \\ &= \mathbf{1}^T \left(\boldsymbol{B}_N - \boldsymbol{Q}_N^T\right)\psi = \mathbf{1}^T \boldsymbol{B}_N \psi.\end{aligned}$$

- Proof requires $\widetilde{\boldsymbol{v}} = \boldsymbol{v}(\widetilde{\boldsymbol{u}})$; the entropy variables $\widetilde{\boldsymbol{v}}$ must be a function of the conservative variables $\widetilde{\boldsymbol{u}}$.

Flux differencing: avoiding the chain rule

- Test with entropy variables $\widetilde{\boldsymbol{v}}$, integrate, and use SBP property:

$$\widetilde{\boldsymbol{v}}^T \left(2\boldsymbol{Q}_N \circ \boldsymbol{F}_S\right)\boldsymbol{1} = \widetilde{\boldsymbol{v}}^T \left(\left(\begin{bmatrix} 0 & \\ & \boldsymbol{W}_f \boldsymbol{n} \end{bmatrix} + \boldsymbol{Q}_N - \boldsymbol{Q}_N^T\right) \circ \boldsymbol{F}_S\right)\boldsymbol{1}.$$

- Only boundary terms appear in final estimate; volume terms become boundary terms using properties of $\left(\boldsymbol{F}_S\right)_{ij} = \boldsymbol{f}_S\left(\widetilde{\boldsymbol{u}}_i, \widetilde{\boldsymbol{u}}_j\right)$

$$\begin{aligned}\widetilde{\boldsymbol{v}}^T \left(\left(\boldsymbol{Q}_N - \boldsymbol{Q}_N^T\right) \circ \boldsymbol{F}_S\right)\boldsymbol{1} &= \widetilde{\boldsymbol{v}}^T \left(\boldsymbol{Q}_N \circ \boldsymbol{F}_S\right)\boldsymbol{1} - \boldsymbol{1}^T \left(\boldsymbol{Q}_N \circ \boldsymbol{F}_S\right)\widetilde{\boldsymbol{v}}\\ &= \boldsymbol{1}^T \left(\boldsymbol{B}_N - \boldsymbol{Q}_N^T\right)\psi = \boldsymbol{1}^T \boldsymbol{B}_N \psi.\end{aligned}$$

- Proof requires $\widetilde{\boldsymbol{v}} = \boldsymbol{v}(\widetilde{\boldsymbol{u}})$; the entropy variables $\widetilde{\boldsymbol{v}}$ must be a function of the conservative variables $\widetilde{\boldsymbol{u}}$.

Modifying the conservative variables

- Conservative variables $\boldsymbol{u}_h$ and test functions are polynomial, but the entropy variables $\boldsymbol{v}(\boldsymbol{u}_h) \notin P^N$!
- Evaluate flux $\boldsymbol{f}_S$ using modified conservative variables $\widetilde{\boldsymbol{u}}$

$$\widetilde{\boldsymbol{u}} = \boldsymbol{u}\left(P_N \boldsymbol{v}(\boldsymbol{u}_h)\right).$$

- If $\boldsymbol{v}(\boldsymbol{u})$ is an invertible mapping, this choice of $\widetilde{\boldsymbol{u}}$ ensures that

$$\widetilde{\boldsymbol{v}} = \boldsymbol{v}(\widetilde{\boldsymbol{u}}) = P_N \boldsymbol{v}(\boldsymbol{u}_h) \in P^N.$$

- Local conservation w.r.t. a generalized Lax-Wendroff theorem.

Shi and Shu (2017). *On local conservation of numerical methods for conservation laws.*

A discretely entropy conservative DG method

Theorem (Chan 2018)

Let $\boldsymbol{u}_h(\boldsymbol{x}) = \sum_j \widehat{\boldsymbol{u}}_j \phi_j(\boldsymbol{x})$ *and* $\widetilde{\boldsymbol{u}} = \boldsymbol{u}\left(P_N \boldsymbol{v}\right)$*. Let* $\widehat{\boldsymbol{u}}$ *locally solve*

$$\frac{\mathrm{d}\widehat{\boldsymbol{u}}}{\mathrm{dt}} + \sum_{i=1}^{d} \left[\begin{array}{cc} \boldsymbol{P}_q & \boldsymbol{L}_f \end{array}\right] \left(2\boldsymbol{D}_N^i \circ \boldsymbol{F}_S^i\right) \boldsymbol{1} + \boldsymbol{L}_f \left(\boldsymbol{f}_S^i(\widetilde{\boldsymbol{u}}^+, \widetilde{\boldsymbol{u}}) - \boldsymbol{f}^i(\widetilde{\boldsymbol{u}})\right) \boldsymbol{n}_i = 0.$$

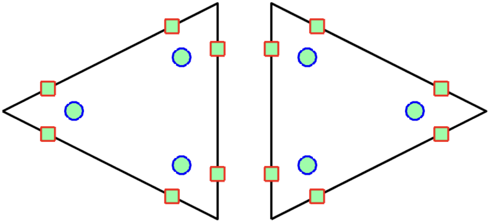
Assuming continuity in time, $\boldsymbol{u}_h(\boldsymbol{x})$ *satisfies the quadrature form of*

$$\int_\Omega \frac{\partial S(\boldsymbol{u}_h)}{\partial t} + \sum_{i=1}^{d} \int_{\partial\Omega} \left(\left(P_N \boldsymbol{v}\right)^T \boldsymbol{f}^i(\widetilde{\boldsymbol{u}}) - \psi_i(\widetilde{\boldsymbol{u}})\right) \boldsymbol{n}_i = 0.$$

- Can modify interface flux (e.g. Lax-Friedrichs or matrix dissipation) to change the entropy equality to an entropy inequality.

Winters, Derigs, Gassner, and Walch (2017). *A uniquely defined entropy stable matrix dissipation operator for high Mach number ideal MHD and compressible Euler simulations.*

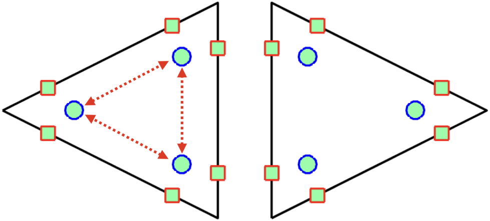
Illustration of main steps of ESDG



$$\left(\underbrace{\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ -\boldsymbol{B}^T & \boldsymbol{C} \end{bmatrix}}_{\boldsymbol{Q}_N^i} \circ \underbrace{\begin{bmatrix} \boldsymbol{F}_S^{vv} & \boldsymbol{F}_S^{vf} \\ \boldsymbol{F}_S^{fv} & \boldsymbol{F}_S^{ff} \end{bmatrix}}_{\boldsymbol{F}_S}\right)\mathbf{1}$$

- Interpolate projected entropy variables $P_N \boldsymbol{v}(\boldsymbol{u})$ to all nodes.
- Compute interactions $\boldsymbol{f}_S(\boldsymbol{u}_L, \boldsymbol{u}_R)$ between volume quadrature nodes.
- Compute interactions between surface nodes of neighboring elements
- Compute interactions between volume/surface nodes.

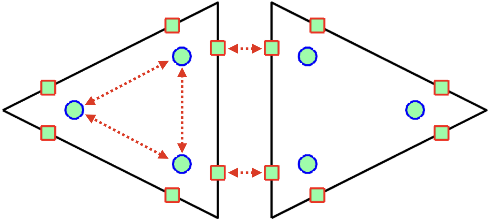
Illustration of main steps of ESDG



$$\left(\underbrace{\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ -\boldsymbol{B}^T & \boldsymbol{C} \end{bmatrix}}_{\boldsymbol{Q}_N^i} \circ \underbrace{\begin{bmatrix} \boldsymbol{F}_S^{vv} & \boldsymbol{F}_S^{vf} \\ \boldsymbol{F}_S^{fv} & \boldsymbol{F}_S^{ff} \end{bmatrix}}_{\boldsymbol{F}_S}\right)\mathbf{1}$$

- Interpolate projected entropy variables $P_N \boldsymbol{v}(\boldsymbol{u})$ to all nodes.
- Compute interactions $\boldsymbol{f}_S(\boldsymbol{u}_L, \boldsymbol{u}_R)$ between volume quadrature nodes.
- Compute interactions between surface nodes of neighboring elements
- Compute interactions between volume/surface nodes.

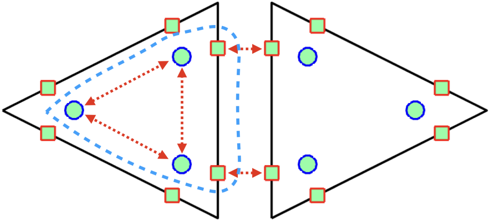
Illustration of main steps of ESDG



$$\left(\underbrace{\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ -\boldsymbol{B}^T & \boldsymbol{C} \end{bmatrix}}_{\boldsymbol{Q}_N^i} \circ \underbrace{\begin{bmatrix} \boldsymbol{F}_S^{vv} & \boldsymbol{F}_S^{vf} \\ \boldsymbol{F}_S^{fv} & \boldsymbol{F}_S^{ff} \end{bmatrix}}_{\boldsymbol{F}_S} \right) \mathbf{1}$$

- Interpolate projected entropy variables $P_N \boldsymbol{v}(\boldsymbol{u})$ to all nodes.
- Compute interactions $\boldsymbol{f}_S(\boldsymbol{u}_L, \boldsymbol{u}_R)$ between volume quadrature nodes.
- Compute interactions between surface nodes of neighboring elements
- Compute interactions between volume/surface nodes.

Illustration of main steps of ESDG



$$\left(\underbrace{\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ -\boldsymbol{B}^T & \boldsymbol{C} \end{bmatrix}}_{\boldsymbol{Q}_N^i} \circ \underbrace{\begin{bmatrix} \boldsymbol{F}_S^{vv} & \boldsymbol{F}_S^{vf} \\ \boldsymbol{F}_S^{fv} & \boldsymbol{F}_S^{ff} \end{bmatrix}}_{\boldsymbol{F}_S} \right) \mathbf{1}$$

- Interpolate projected entropy variables $P_N \boldsymbol{v}(\boldsymbol{u})$ to all nodes.
- Compute interactions $\boldsymbol{f}_S(\boldsymbol{u}_L, \boldsymbol{u}_R)$ between volume quadrature nodes.
- Compute interactions between surface nodes of neighboring elements
- Compute interactions between volume/surface nodes.

Talk outline

1 Stability of high order DG: linear vs nonlinear PDEs

2 Summation by parts finite differences and high order DG

3 Entropy stable formulations and flux differencing

4 Numerical experiments
- 1D experiments
- Triangular and tetrahedral meshes
- Quadrilateral and hexahedral meshes

Talk outline

1 Stability of high order DG: linear vs nonlinear PDEs

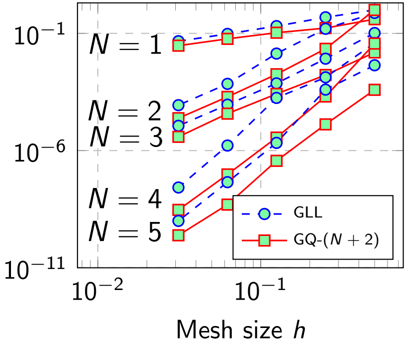
2 Summation by parts finite differences and high order DG

3 Entropy stable formulations and flux differencing

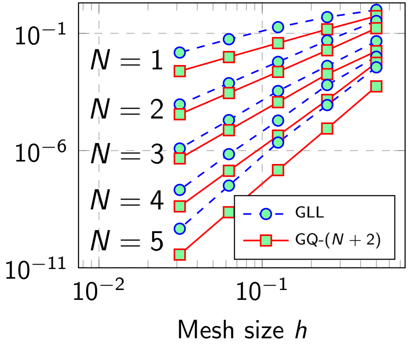
4 Numerical experiments
- 1D experiments
- Triangular and tetrahedral meshes
- Quadrilateral and hexahedral meshes

1D compressible Euler equations

- Inexact Gauss-Legendre-Lobatto (GLL) vs Gauss (GQ) quadratures.
- Entropy conservative (EC) and dissipative Lax-Friedrichs (LF) fluxes.
- No additional stabilization, filtering, or limiting.



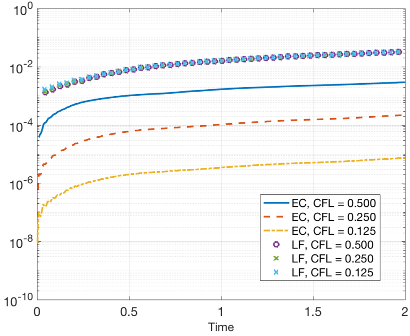
(a) Entropy conservative flux



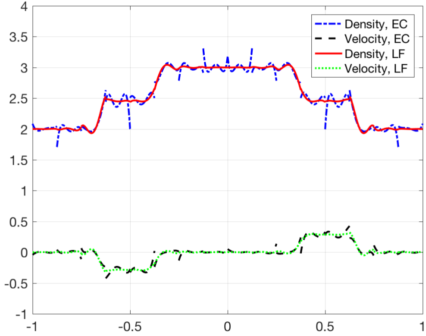
(b) With Lax-Friedrichs penalization

Conservation of entropy: fully discrete schemes

- Entropy conservation: *semi-discrete*, not fully discrete.
- $\Delta S(\boldsymbol{u}) = |S(\boldsymbol{u}(x,t)) - S(\boldsymbol{u}(x,0))| \to 0$ as as $\Delta t \to 0$.



(a) $\Delta S(\boldsymbol{u})$ for various Δt

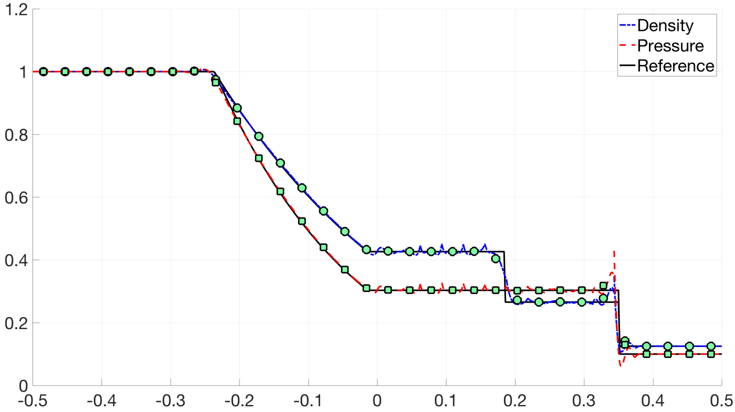


(b) $\rho(x), u(x)$ $(N = 4, K = 16)$

Solution and change in entropy $\Delta S(\boldsymbol{u})$ for entropy conservative (EC) and Lax-Friedrichs (LF) fluxes (using GQ-$(N+2)$ quadrature).

1D Sod shock tube

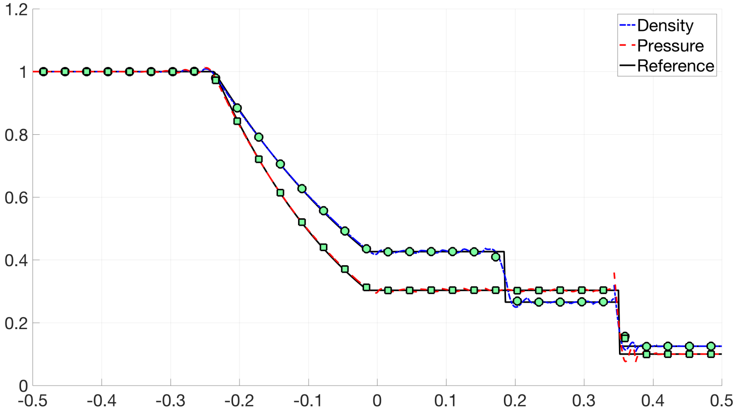
- Circles are cell averages.
- CFL of .125 used for both GLL-$(N+1)$and GQ-$(N+2)$.



$N = 4$, $K = 32$, $(N+1)$ point Gauss-Lobatto-Legendre quadrature.

1D Sod shock tube

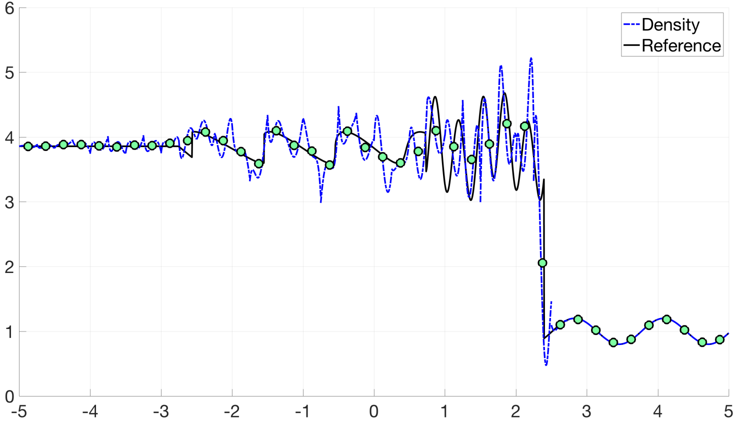
- Circles are cell averages.
- CFL of .125 used for both GLL-$(N+1)$and GQ-$(N+2)$.



$N=4$, $K=32$, $(N+2)$ point Gauss quadrature.

1D sine-shock interaction

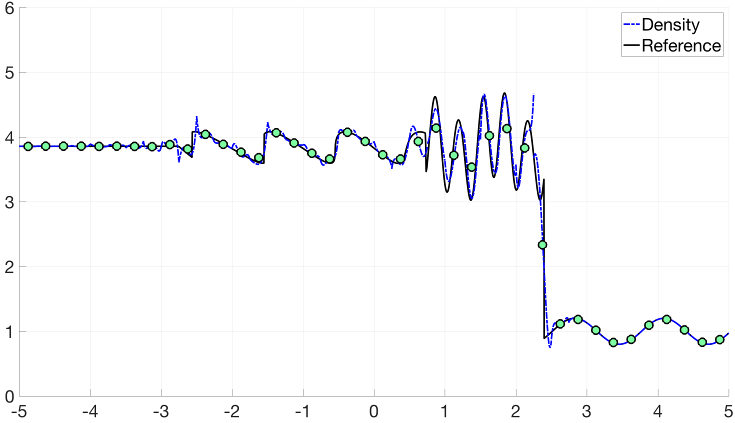
- GQ-$(N+2)$ needs smaller CFL (.05 vs .125) for stability.



$N = 4$, $K = 40$, $CFL = .05$, $(N+1)$ point Gauss-Lobatto-Legendre quadrature.

1D sine-shock interaction

- GQ-$(N+2)$ needs smaller CFL (.05 vs .125) for stability.



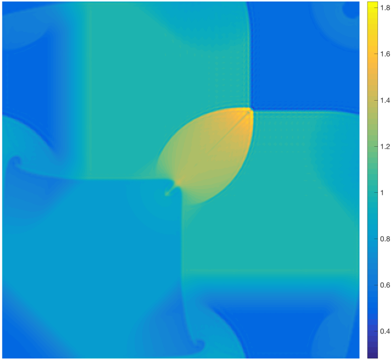
$N = 4$, $K = 40$, $CFL = .05$, $(N+2)$ point Gauss quadrature.

Talk outline

1. Stability of high order DG: linear vs nonlinear PDEs
2. Summation by parts finite differences and high order DG
3. Entropy stable formulations and flux differencing
4. Numerical experiments
   - 1D experiments
   - Triangular and tetrahedral meshes
   - Quadrilateral and hexahedral meshes

2D Riemann problem

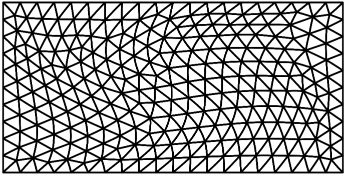
- Uniform 64×64 mesh: $N = 3$, CFL .125, Lax-Friedrichs stabilization.
- No limiting or artificial viscosity required to maintain stability!
- Periodic on larger domain ("natural" boundary conditions unstable).



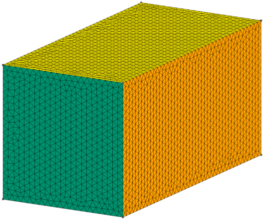
(a) $\Omega = [-1, 1]^2$

(b) $\Omega = [-.5, .5]^2$, 32×32 elements

Smooth isentropic vortex and curved meshes in 2D/3D



(a) 2D triangular mesh



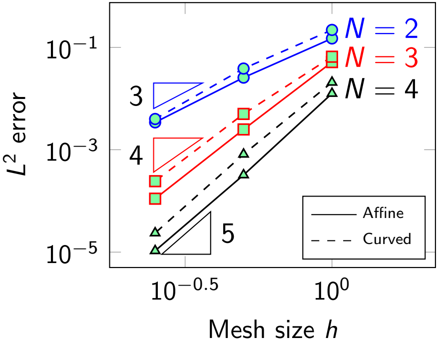
(b) 3D tetrahedral mesh

Figure: Example of 2D and 3D meshes used for convergence experiments.

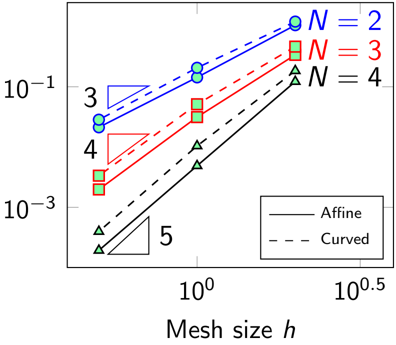
- Entropy stability: needs discrete geometric conservation law (GCL).
- Generalized mass lumping: weight-adjusted mass matrices.
- Modify $\widetilde{\boldsymbol{u}} = \boldsymbol{u}(\widetilde{\boldsymbol{v}})$, $\widetilde{\boldsymbol{v}} = \widetilde{P}_N^k \boldsymbol{v}(\boldsymbol{u}_h)$ using weight-adjusted projection $\widetilde{P}_N^k$.

Visbal and Gaitonde (2002). On the Use of Higher-Order Finite-Difference Schemes on Curvilinear and Deforming Meshes.
Kopriva (2006). Metric identities and the discontinuous spectral element method on curvilinear meshes.
Chan, Hewett, and Warburton (2016). *Weight-adjusted discontinuous Galerkin methods: curvilinear meshes.*

Smooth isentropic vortex and curved meshes in 2D/3D



(a) 2D results



(b) 3D results

L^2 errors for 2D/3D isentropic vortex at $T = 5$ on affine, curved meshes.

Visbal and Gaitonde (2002). On the Use of Higher-Order Finite-Difference Schemes on Curvilinear and Deforming Meshes.
Kopriva (2006). Metric identities and the discontinuous spectral element method on curvilinear meshes.
Chan, Hewett, and Warburton (2016). *Weight-adjusted discontinuous Galerkin methods: curvilinear meshes.*

Taylor-Green vortex

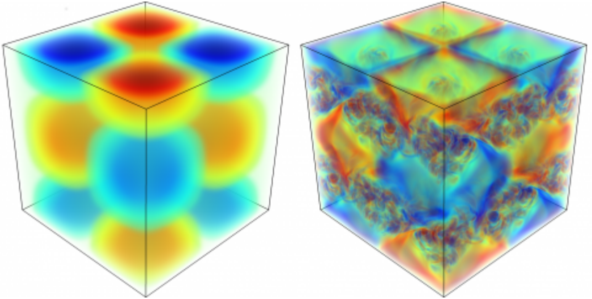
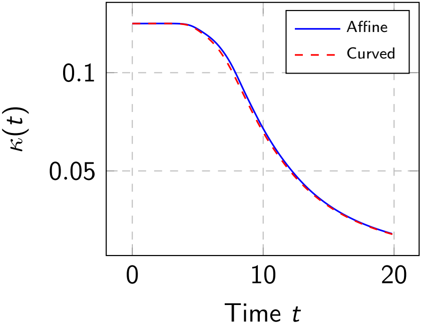


Figure: Isocontours of z-vorticity for Taylor-Green at $t = 0, 10$ seconds.

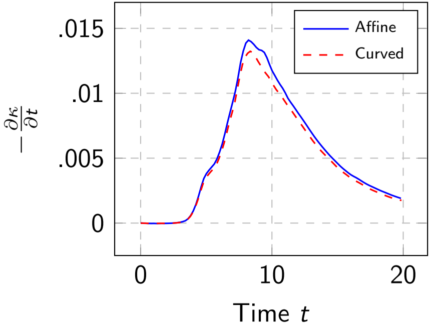
- Simple turbulence-like behavior (generation of small scales).
- Inviscid Taylor-Green: tests robustness w.r.t. under-resolved solutions.

https://how4.cenaero.be/content/bs1-dns-taylor-green-vortex-re1600.

Taylor-Green vortex: kinetic energy dissipation rate



(a) Kinetic energy



(b) KE dissipation rate

Figure: Evolution of kinetic energy $\kappa(t)$ and kinetic energy dissipation rate $-\frac{\partial \kappa}{\partial t}$ for $N = 3, h = \pi/8, \mathrm{CFL} = .25$ on affine and curved meshes.

Taylor-Green vortex: kinetic energy dissipation rate

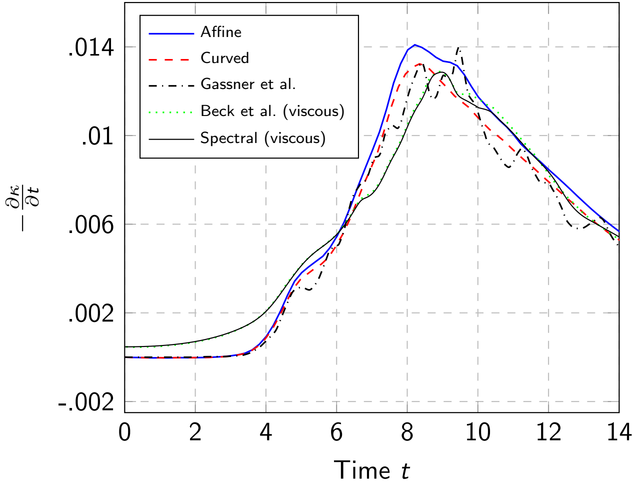


Figure: Evolution of kinetic energy $\kappa(t)$ and kinetic energy dissipation rate $-\frac{\partial \kappa}{\partial t}$ for $N = 3, h = \pi/8, \mathrm{CFL} = .25$ on affine and curved meshes.

Talk outline

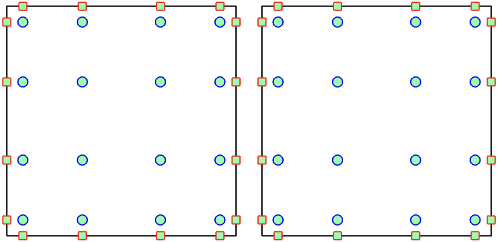
1 Stability of high order DG: linear vs nonlinear PDEs

2 Summation by parts finite differences and high order DG

3 Entropy stable formulations and flux differencing

4 Numerical experiments
- 1D experiments
- Triangular and tetrahedral meshes
- Quadrilateral and hexahedral meshes

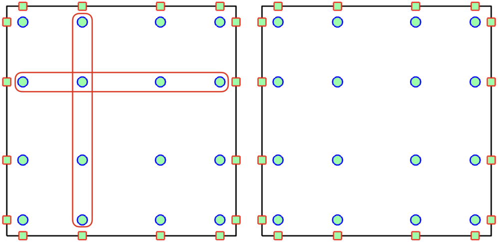
Entropy stable Gauss collocation: main steps



$$\left(\underbrace{\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ -\boldsymbol{B}^T & \boldsymbol{C} \end{bmatrix}}_{\boldsymbol{Q}_N^i} \circ \underbrace{\begin{bmatrix} \boldsymbol{F}_S^{vv} & \boldsymbol{F}_S^{vf} \\ \boldsymbol{F}_S^{fv} & \boldsymbol{F}_S^{ff} \end{bmatrix}}_{\boldsymbol{F}_S} \right) \mathbf{1}$$

- Advantage over tetrahedral elements: tensor product structure.
- Reduces computational costs from $O(N^6)$ to $O(N^4)$ in 3D.
- New approach: collocate at Gauss nodes instead of GLL nodes.

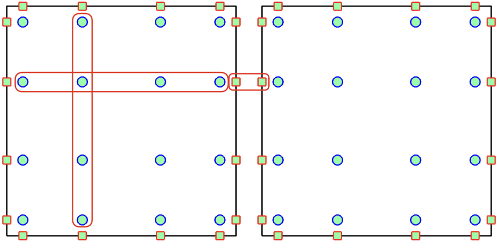
Entropy stable Gauss collocation: main steps



$$\left(\underbrace{\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ -\boldsymbol{B}^T & \boldsymbol{C}\end{bmatrix}}_{\boldsymbol{Q}_N^i} \circ \underbrace{\begin{bmatrix} \boldsymbol{F}_S^{vv} & \boldsymbol{F}_S^{vf}\\ \boldsymbol{F}_S^{fv} & \boldsymbol{F}_S^{ff}\end{bmatrix}}_{\boldsymbol{F}_S}\right)\mathbf{1}$$

- Advantage over tetrahedral elements: tensor product structure.
- Reduces computational costs from $O(N^6)$ to $O(N^4)$ in 3D.
- New approach: collocate at Gauss nodes instead of GLL nodes.

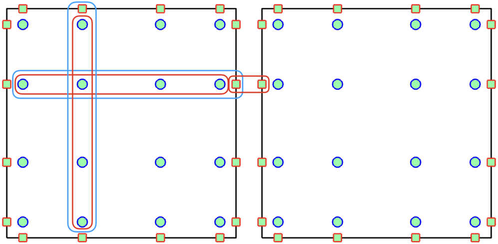
Entropy stable Gauss collocation: main steps



$$\left(\underbrace{\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ -\boldsymbol{B}^T & \boldsymbol{C} \end{bmatrix}}_{\boldsymbol{Q}_N^i} \circ \underbrace{\begin{bmatrix} \boldsymbol{F}_S^{vv} & \boldsymbol{F}_S^{vf} \\ \boldsymbol{F}_S^{fv} & \boldsymbol{F}_S^{ff} \end{bmatrix}}_{\boldsymbol{F}_S}\right)\mathbf{1}$$

- Advantage over tetrahedral elements: tensor product structure.
- Reduces computational costs from $O(N^6)$ to $O(N^4)$ in 3D.
- New approach: collocate at Gauss nodes instead of GLL nodes.

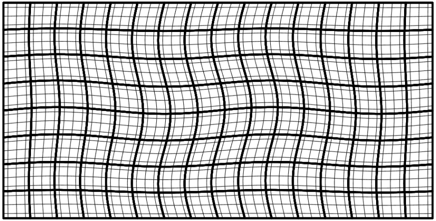
Entropy stable Gauss collocation: main steps



$$\left(\underbrace{\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ -\boldsymbol{B}^T & \boldsymbol{C} \end{bmatrix}}_{\boldsymbol{Q}_N^i} \circ \underbrace{\begin{bmatrix} \boldsymbol{F}_S^{vv} & \boldsymbol{F}_S^{vf} \\ \boldsymbol{F}_S^{fv} & \boldsymbol{F}_S^{ff} \end{bmatrix}}_{\boldsymbol{F}_S}\right)\mathbf{1}$$

- Advantage over tetrahedral elements: tensor product structure.
- Reduces computational costs from $O(N^6)$ to $O(N^4)$ in 3D.
- New approach: collocate at Gauss nodes instead of GLL nodes.

Improved errors on curved meshes



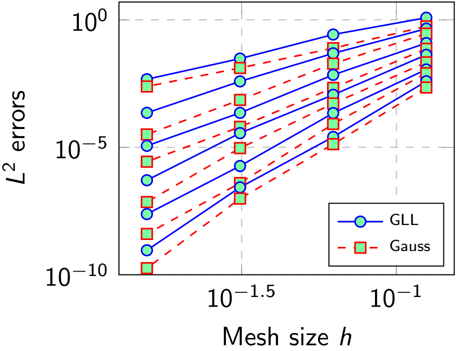
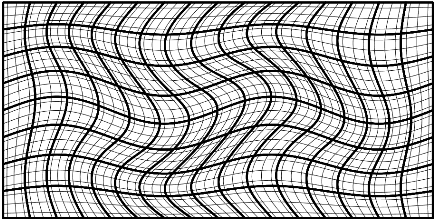


Figure: L^2 errors for the 2D isentropic vortex at time $T = 5$ for degree $N = 2, \ldots, 7$ GLL and Gauss collocation schemes (similar behavior in 3D).

Improved errors on curved meshes



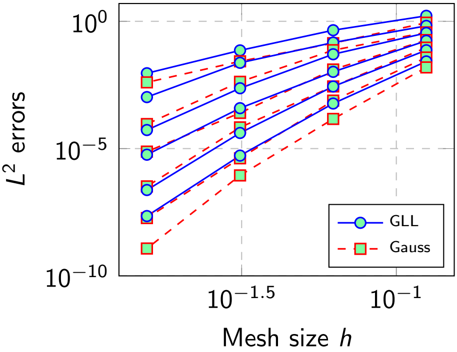
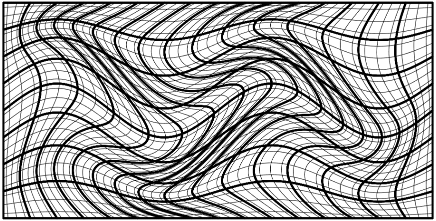


Figure: L^2 errors for the 2D isentropic vortex at time $T = 5$ for degree $N = 2, \ldots, 7$ GLL and Gauss collocation schemes (similar behavior in 3D).

Improved errors on curved meshes



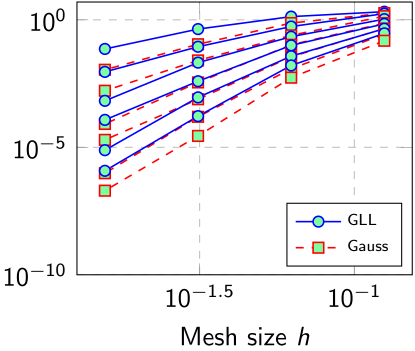
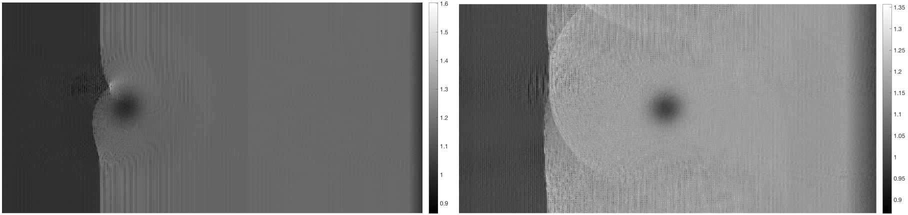


Figure: L^2 errors for the 2D isentropic vortex at time $T = 5$ for degree $N = 2, \ldots, 7$ GLL and Gauss collocation schemes (similar behavior in 3D).

Shock vortex interaction

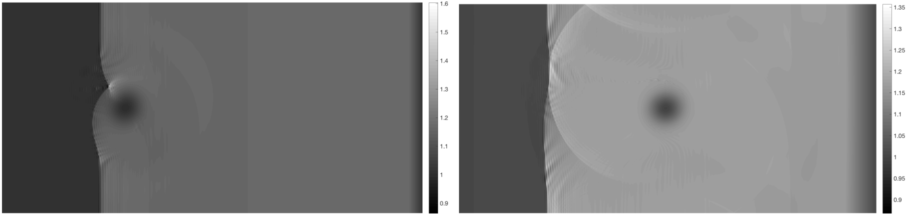


(a) Entropy conservative flux, $T = .3$ (b) Entropy conservative flux, $T = .7$

Figure: Shock vortex interaction problem using high order entropy stable Gauss collocation schemes with $N = 4, h = 1/100$.

Winters, Derigs, Gassner, and Walch (2017). *A uniquely defined entropy stable matrix dissipation operator for high Mach number ideal MHD and compressible Euler simulations.*

Shock vortex interaction



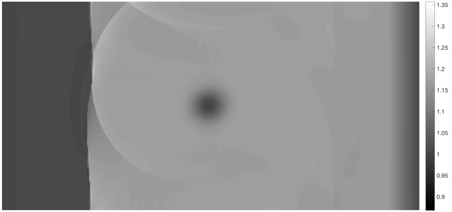
(a) Lax-Friedrichs flux, $T = .3$ (b) Lax-Friedrichs flux, $T = .7$

Figure: Shock vortex interaction problem using high order entropy stable Gauss collocation schemes with $N = 4, h = 1/100$.

Winters, Derigs, Gassner, and Walch (2017). *A uniquely defined entropy stable matrix dissipation operator for high Mach number ideal MHD and compressible Euler simulations.*

Shock vortex interaction

(a) Matrix dissipation flux, $T = .3$

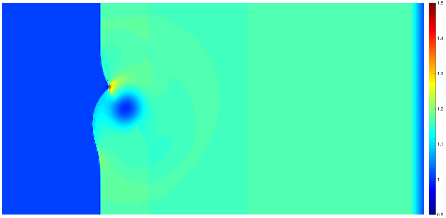


(b) Matrix dissipation flux, $T = .7$

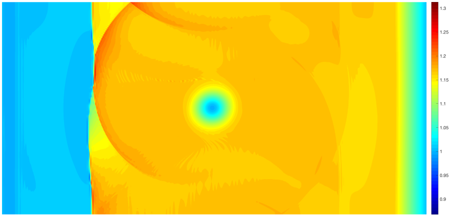
Figure: Shock vortex interaction problem using high order entropy stable Gauss collocation schemes with $N = 4, h = 1/100$.

Winters, Derigs, Gassner, and Walch (2017). *A uniquely defined entropy stable matrix dissipation operator for high Mach number ideal MHD and compressible Euler simulations.*

Shock vortex interaction



(a) Matrix dissipation flux, $T = .3$



(b) Matrix dissipation flux, $T = .7$

Figure: Shock vortex interaction problem using high order entropy stable Gauss collocation schemes with $N = 4, h = 1/100$.

Winters, Derigs, Gassner, and Walch (2017). *A uniquely defined entropy stable matrix dissipation operator for high Mach number ideal MHD and compressible Euler simulations.*

Summary and future work

- Entropy stable high order discontinuous Galerkin methods: semi-discrete stability, improved robustness.
- Additional work required for strong shocks, positivity preservation.
- Currently: hybrid + non-conforming meshes, multi-GPU.
- This work is supported by DMS-1719818 and DMS-1712639.

Thank you! Questions?

Chan, Del Rey Fernandez, Carpenter (2018). *Efficient entropy stable Gauss collocation methods.*

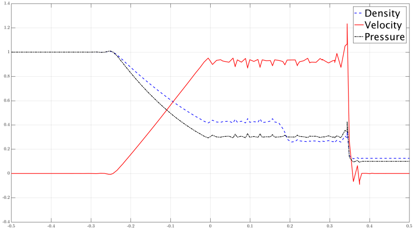
Chan, Wilcox (2018). *On discretely entropy stable weight-adjusted DG methods: curvilinear meshes.*

Chan, Hewett, and Warburton (2016). *Weight-adjusted discontinuous Galerkin methods: curvilinear meshes.*

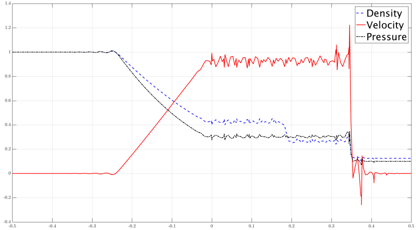
Chan (2017). *On discretely entropy conservative and entropy stable discontinuous Galerkin methods.*

Additional slides

Over-integration is ineffective without L^2 projection



(a) $(N+1)$ points

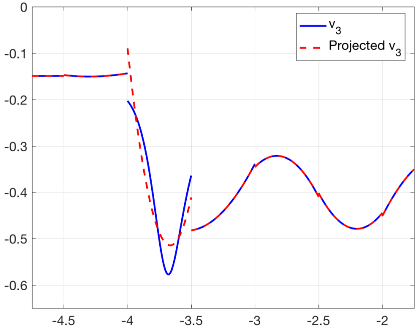


(b) $(N+4)$ points

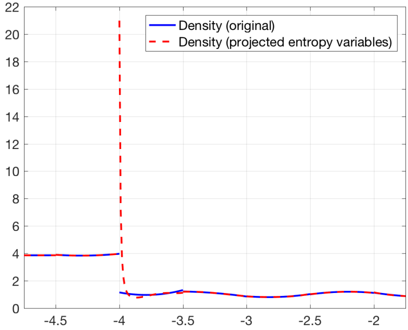
Figure: Numerical results for the Sod shock tube for $N = 4$ and $K = 32$ elements. Over-integrating by increasing the number of quadrature points does not improve solution quality.

On CFL restrictions

- For GLL-$(N+1)$ quadrature, $\widetilde{\boldsymbol{u}} = \boldsymbol{u}\left(P_N \boldsymbol{v}\right) = \boldsymbol{u}$ at GLL points.
- For GQ-$(N+2)$, discrepancy between L^2 projection and interpolation.
- Still need positivity of thermodynamic quantities for stability!



(a) $v_3(x), \left(P_N v_3\right)(x)$



(b) $\rho(x), \rho\left(\left(P_N \boldsymbol{v}\right)(x)\right)$

High order DG on many-core (GPU) architectures

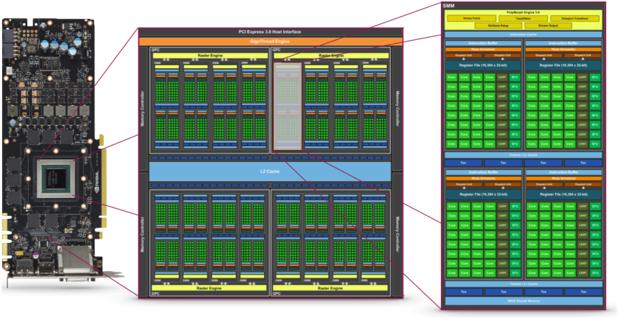


Figure: NVIDIA Maxwell GM204 GPU: 16 cores, 4 SIMD clusters of 32 units.

- Thousands of processing units organized in synchronized groups.
- No free lunch: memory costs (accesses, transfer, latency, storage).

Klockner, Warburton, Bridge, Hesthaven 2009, Nodal discontinuous Galerkin methods on graphics processors.

High order DG on many-core (GPU) architectures

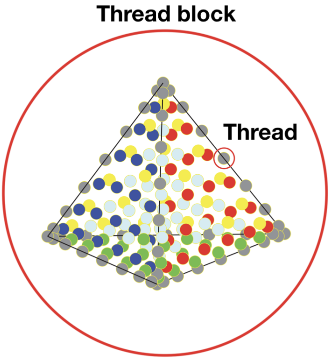


Figure: Thread blocks process elements, threads process degrees of freedom.

- Thousands of processing units organized in synchronized groups.
- No free lunch: memory costs (accesses, transfer, latency, storage).

Klockner, Warburton, Bridge, Hesthaven 2009, Nodal discontinuous Galerkin methods on graphics processors.

High order DG on many-core (GPU) architectures

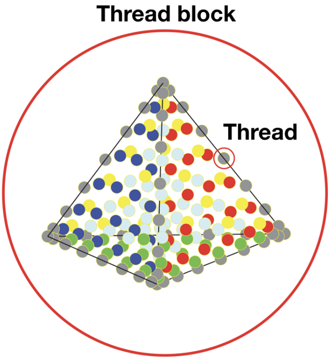


Figure: Thread blocks process elements, threads process degrees of freedom.

- Thousands of processing units organized in synchronized groups.
- No free lunch: memory costs (accesses, transfer, latency, storage).

Klockner, Warburton, Bridge, Hesthaven 2009, Nodal discontinuous Galerkin methods on graphics processors.

Implementing high order entropy stable DG on GPUs

- "FLOPS are free, **but** . . ."

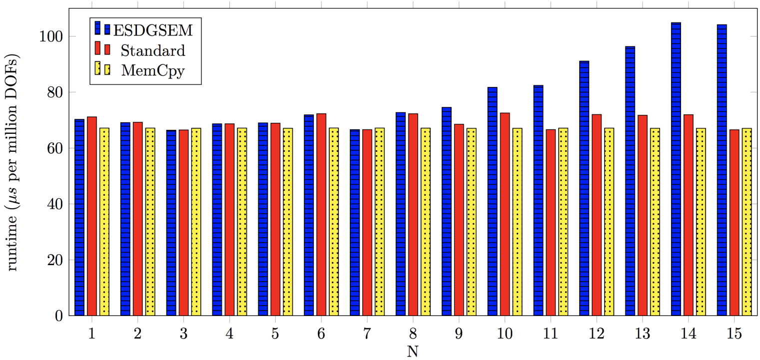
  (bytes are expensive) / (memory is dear) / (postage is extra)

- Standard considerations: minimize CPU-GPU transfers, structured data layouts, reduce global memory accesses, maximize data reuse.

- Arithmetic vs memory latency: need roughly $O(10)$ operations per byte of memory accessed (high arithmetic intensity).

- Standard mat-vec: only $1/10 - 1/2$ FLOPS per byte!

GPUs and flux differencing: when FLOPS are free



- High arithmetic intensity: compute while waiting for global memory.
- On GPUs, extra operations don't increase runtime until $N \geq 9$!

Wintermeyer, Winters, Gassner, Warburton (2018). *An entropy stable discontinuous Galerkin method for the shallow water equations on curvilinear meshes with wet/dry fronts accelerated by GPUs.*