Entropy stable high order discontinuous Galerkin methods for nonlinear conservation laws

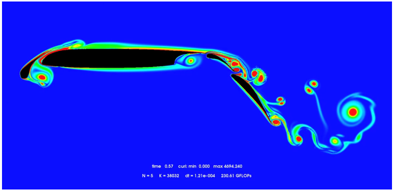
Jesse Chan

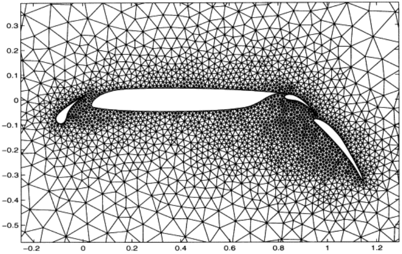
[1]Department of Computational and Applied Mathematics

Department of Mathematics, Rensselaer Polytechnic Institute
October 22, 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).

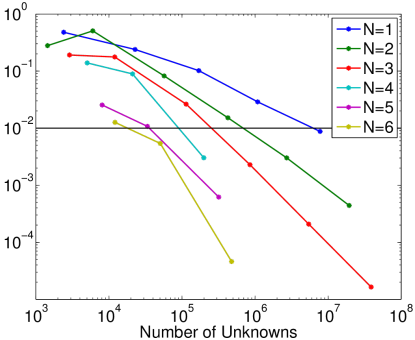




Mesh from Slawig 2001.

High order finite element methods for hyperbolic PDEs

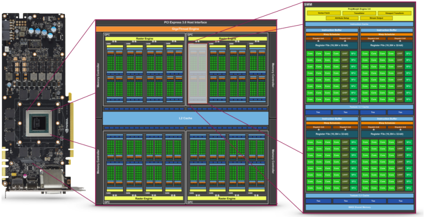
- Focus: high accuracy computational mechanics on complex geometries.
- Applications in fluid dynamics (waves, vorticular flows, turbulence, shocks).
- High order approximations are more accurate per unknown.
- High performance computing on many-core architectures (efficient explicit time-stepping).



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

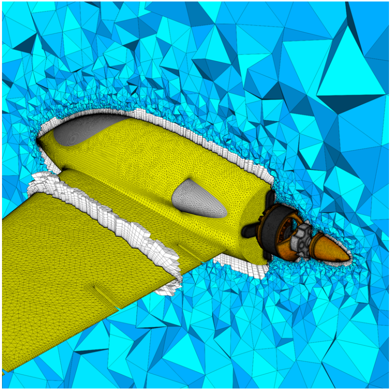
High order finite element methods for hyperbolic PDEs

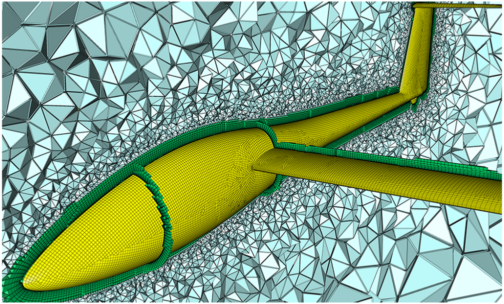
- Focus: high accuracy computational mechanics on complex geometries.
- Applications in fluid dynamics (waves, vorticular flows, turbulence, shocks).
- High order approximations are more accurate per unknown.
- High performance computing on many-core architectures (efficient explicit time-stepping).



Schematic of an NVIDIA graphics processing unit (GPU).

Why FEM? Theory on general unstructured meshes

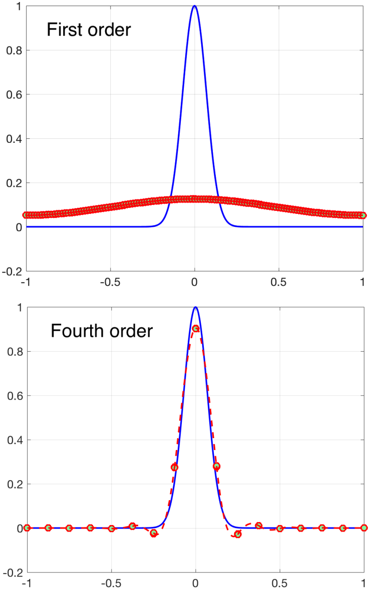




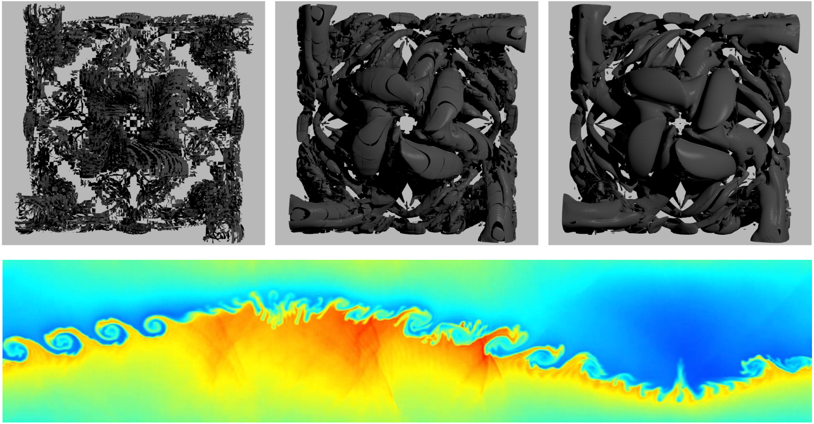
DG methods are compatible with unstructured meshes containing different types of elements (tetrahedra, hexahedra most common, but also prisms and pyramids).

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

Why high order? Low numerical dissipation



Why high order? Low 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 and high order DG
3. Entropy stable formulations and flux differencing
4. Numerical experiments
    - Triangular and tetrahedral meshes
    - Quadrilateral and hexahedral meshes
    - Hybrid and non-conforming meshes

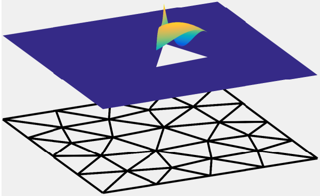
Talk outline

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

Basics of discontinuous Galerkin methods

Discontinuous Galerkin (DG) methods:

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



- Continuous PDE (example: advection)

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0.$$

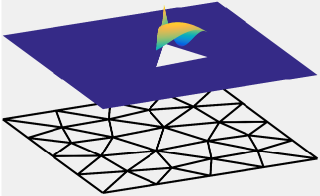
- Local DG form with numerical flux u^*: find $u \in P^N\left(D^k\right)$ such that

$$\int_{D_k} \left(\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x}\right)\phi + \int_{\partial D_k} n_x \left(u^* - u\right)\phi = 0, \qquad \forall \phi \in P^N\left(D^k\right).$$

Basics of discontinuous Galerkin methods

Discontinuous Galerkin (DG) methods:

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

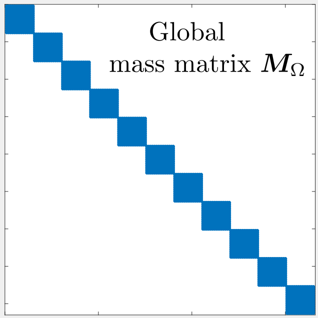


Discretizing in space yields system of ODEs

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

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

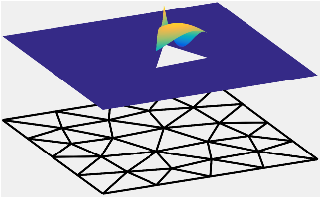
Goal: ensure ODE system is stable in time.



Basics of discontinuous Galerkin methods

Discontinuous Galerkin (DG) methods:

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

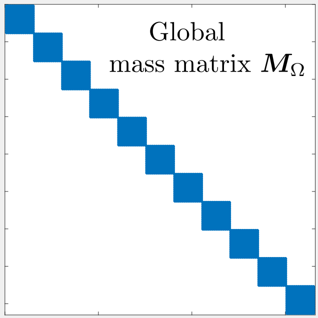


Discretizing in space yields system of ODEs

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

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

Goal: ensure ODE system is stable in time.



DG is semi-discretely energy stable for linear advection

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

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

- DG numerical "penalty" flux, where $[\![u]\!] = u^+ - u$ and $\tau \geq 0$.

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

- 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 $[-1,1]$

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

- DG numerical "penalty" flux, where $[\![u]\!] = u^+ - u$ and $\tau \geq 0$.

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

- 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 $[-1, 1]$

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

- DG numerical "penalty" flux, where $[\![u]\!] = u^+ - u$ and $\tau \geq 0$.

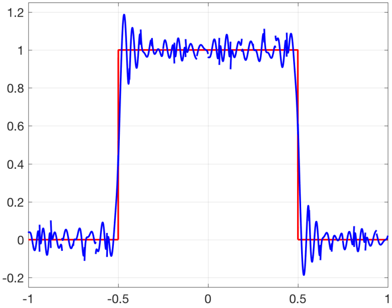
$$\sum_k \int_{D^k} \left(\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x}\right) v \,\mathrm{d}x + \frac{1}{2}\int_{\partial D^k} \left([\![u]\!] n_x + \tau [\![u]\!]\right) v \,\mathrm{d}x = 0.$$

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

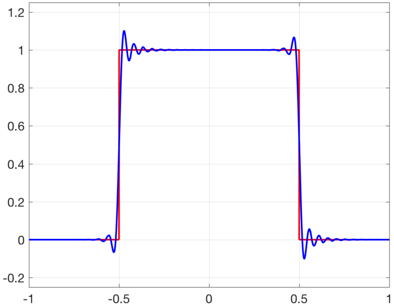
$$\sum_k \frac{\partial}{\partial t} \|u\|^2_{D^k} \leq -\sum_k \frac{\tau}{2} \int_{\partial D^k} [\![u]\!]^2 \,\mathrm{d}x.$$

Energy conservative vs. energy stable DG methods

- Energy estimate implies that $\|u\|$ is non-increasing for $\tau \geq 0$.
- 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).

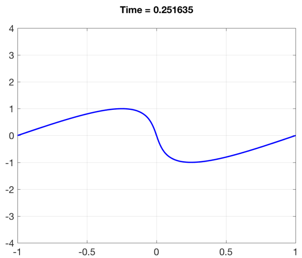
$$\frac{\partial \boldsymbol{u}}{\partial t} + \frac{\partial \boldsymbol{f}(\boldsymbol{u})}{\partial x} = 0.$$

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

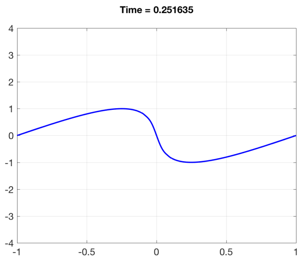
$$\int_\Omega \boldsymbol{v}^T \left(\frac{\partial \boldsymbol{u}}{\partial t} + \frac{\partial \boldsymbol{f}(\boldsymbol{u})}{\partial x}\right) = 0, \qquad \boldsymbol{v} = \frac{\partial S}{\partial \boldsymbol{u}}$$
$$\Longrightarrow \int_\Omega \frac{\partial S(\boldsymbol{u})}{\partial t} + \left(\boldsymbol{v}^T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

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

Why are discretizations of nonlinear PDEs unstable?



(a) Exact solution



(b) 8th order DG

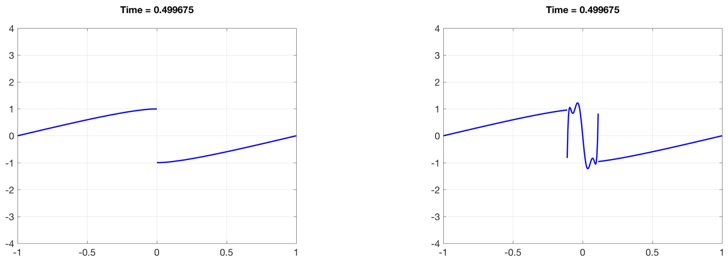
- Burgers' equation: $f(u) = u^2/2$. How to compute $\frac{\partial}{\partial x} f(u)$?

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

- 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) Exact solution (b) 8th order DG

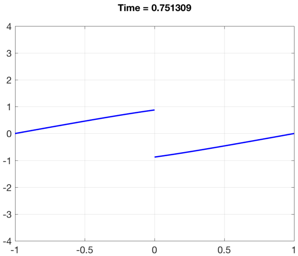
- Burgers' equation: $f(u) = u^2/2$. How to compute $\frac{\partial}{\partial x} f(u)$?

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

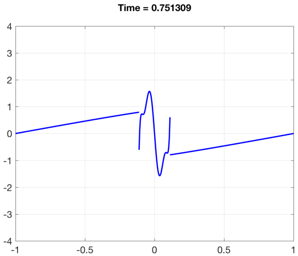
- Differentiating L^2 projection P_N + inexact quadrature: no chain rule.

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

Why are discretizations of nonlinear PDEs unstable?



(a) Exact solution



(b) 8th order DG

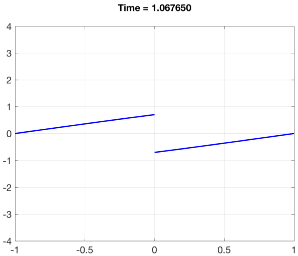
- Burgers' equation: $f(u) = u^2/2$. How to compute $\frac{\partial}{\partial x} f(u)$?

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

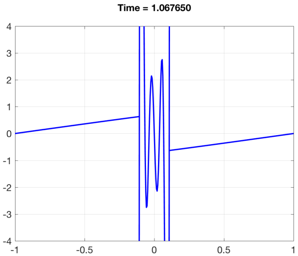
- Differentiating L^2 projection P_N + inexact quadrature: no chain rule.

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

Why are discretizations of nonlinear PDEs unstable?



(a) Exact solution



(b) 8th order DG

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

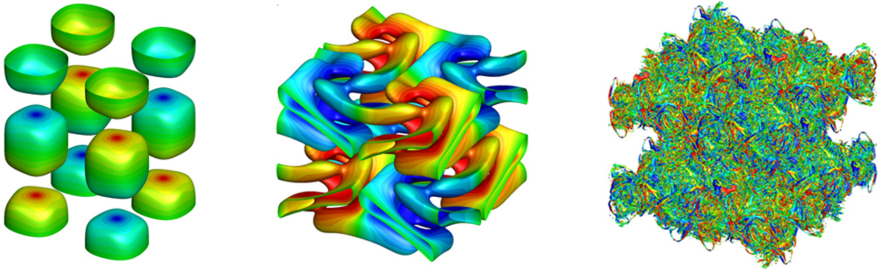
$$\frac{\partial u}{\partial t} + \frac{1}{2}\frac{\partial u^2}{\partial x} = 0, \qquad u \in P^N(D^k), \quad u^2 \notin P^N(D^k).$$

- 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 between high order accuracy vs stability

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

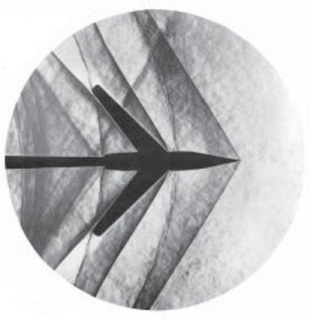


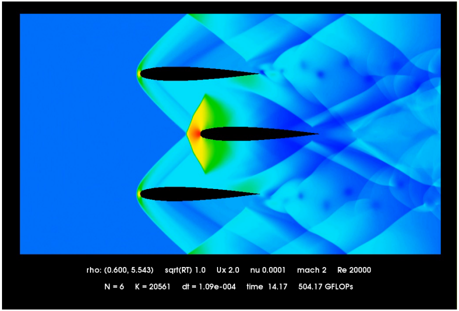
Under-resolved solutions: turbulence (inviscid Taylor-Green vortex).

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

Tradeoff between high order accuracy vs stability

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



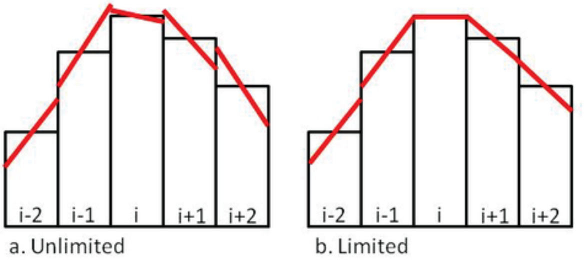


Under-resolved solutions: shock waves.

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

Tradeoff between high order accuracy vs stability

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

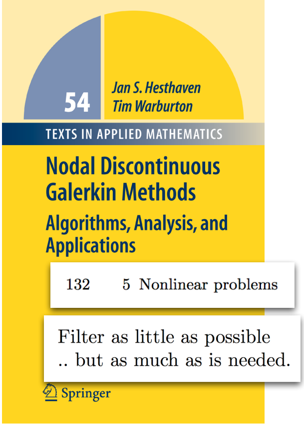


Slope limiting for a finite volume method.

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

Tradeoff between high order accuracy vs stability

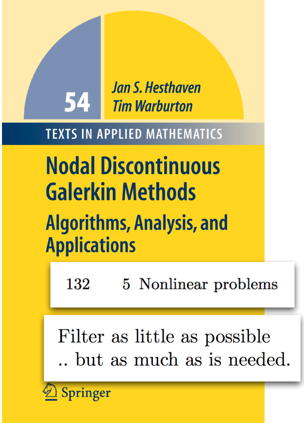
- Asymptotic stability for smooth solutions (not shocks or turbulence!)
- Common fix: stabilize by regularizing (limiters, filters, art. viscosity).

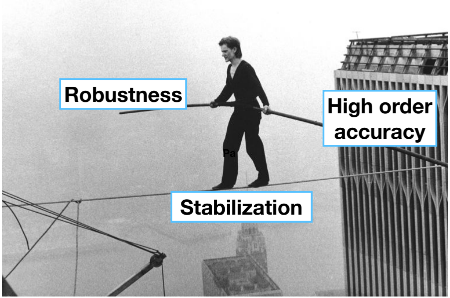


Figures courtesy of Gregor Gassner, T. Warburton, Coastal Inlets Research Program (CIRP), "Man on Wire" (2008).

Tradeoff between high order accuracy vs stability

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



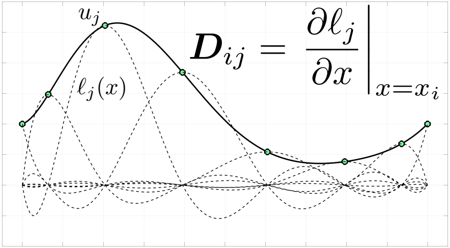


Figures courtesy of Gregor Gassner, T. Warburton, Coastal Inlets Research Program (CIRP), “Man on Wire” (2008).

Talk outline

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

Nodal DG and summation-by-parts (SBP) in 1D



$$\boldsymbol{B} = \begin{bmatrix} -1 & & & \\ & 0 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}$$

- Gauss-Legendre-Lobatto (GLL) quadrature + nodal basis.
- Mimic integration by parts algebraically using differentiation matrix $\boldsymbol{D}$, diagonal (lumped) mass matrix $\boldsymbol{M}$, and boundary matrix $\boldsymbol{B}$

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

Gassner (2013). *A skew-symmetric DG-SEM discretization and its relation to SBP-SAT finite difference methods.*

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 $\boldsymbol{u}^T\boldsymbol{M}$, note $\boldsymbol{Q} = \boldsymbol{M}\boldsymbol{D}$. Use that 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

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

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

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

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 $\boldsymbol{u}^T\boldsymbol{M}$, note $\boldsymbol{Q} = \boldsymbol{M}\boldsymbol{D}$. Use that 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

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

- Semi-discrete stability: multiply by $\boldsymbol{u}^T\boldsymbol{M}$, note $\boldsymbol{Q} = \boldsymbol{M}\boldsymbol{D}$. Use that 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 $\boldsymbol{u}^T\boldsymbol{M}$, note $\boldsymbol{Q} = \boldsymbol{M}\boldsymbol{D}$. Use that 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

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

- Semi-discrete stability: multiply by $\boldsymbol{u}^T\boldsymbol{M}$, note $\boldsymbol{Q} = \boldsymbol{M}\boldsymbol{D}$. Use that 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

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

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

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

Revisiting Burgers' equation: stable split formulations

- Rewrite Burgers' equation in split form

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

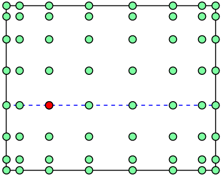
- SBP discretization of split formulation

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

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

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

Current entropy stable SBP discretizations

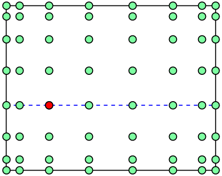


(a) GLL collocation

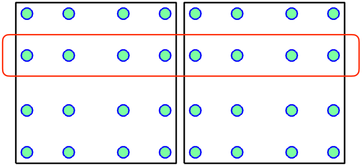
- Current: build SBP matrices using quadrature with boundary nodes.
- Gauss quadrature: more accurate but expensive coupling conditions.
- Tetrahedra, wedges, pyramids? Non-polynomials? Over-integration?

Carpenter et al. (2014), Gassner, Winters, and Kopriva (2016), Hicken et al. (2016), Crean et al. (2018)

Current entropy stable SBP discretizations



(a) GLL collocation

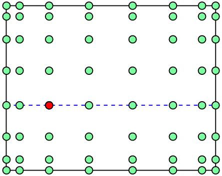


(b) Coupling for Gauss nodes

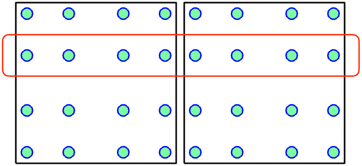
- Current: build SBP matrices using quadrature with boundary nodes.
- Gauss quadrature: more accurate but expensive coupling conditions.
- Tetrahedra, wedges, pyramids? Non-polynomials? Over-integration?

Carpenter et al. (2014), Gassner, Winters, and Kopriva (2016), Hicken et al. (2016), Crean et al. (2018)

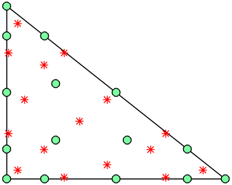
Current entropy stable SBP discretizations



(a) GLL collocation



(b) Coupling for Gauss nodes

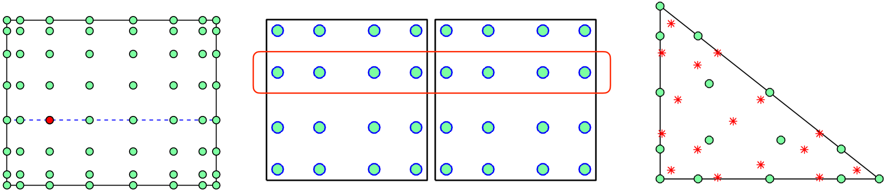


(c) Nodes vs quadrature

- Current: build SBP matrices using quadrature with boundary nodes.
- Gauss quadrature: more accurate but expensive coupling conditions.
- Tetrahedra, wedges, pyramids? Non-polynomials? Over-integration?

Carpenter et al. (2014), Gassner, Winters, and Kopriva (2016), Hicken et al. (2016), Crean et al. (2018)

Current entropy stable SBP discretizations



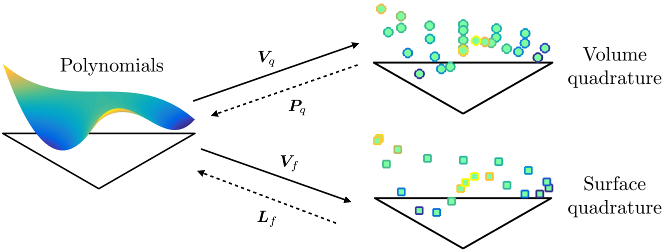
(a) GLL collocation (b) Coupling for Gauss nodes (c) Nodes vs quadrature

- Current: build SBP matrices using quadrature with boundary nodes.
- Gauss quadrature: more accurate but expensive coupling conditions.
- Tetrahedra, wedges, pyramids? Non-polynomials? Over-integration?

Goal: entropy stable high order DG with compact coupling using arbitrary basis functions and general quadrature rules.

Carpenter et al. (2014), Gassner, Winters, and Kopriva (2016), Hicken et al. (2016), Crean et al. (2018)

Polynomial bases and quadrature-based matrices



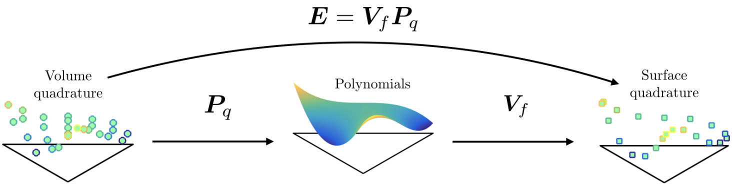
- Assume degree $2N$ volume, surface quadratures $(\boldsymbol{x}_i^q, \boldsymbol{w}_i^q)$, $(\boldsymbol{x}_i^f, \boldsymbol{w}_i^f)$, and basis $\phi_1, \ldots, \phi_{N_p}$. Define interpolation matrices $\boldsymbol{V}_q, \boldsymbol{V}_f$

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

- 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 “finite difference” matrices



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

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

- Generalized summation-by-parts: let $\boldsymbol{Q}_i = \boldsymbol{W}\boldsymbol{D}_q^i$ and $\boldsymbol{E} = \boldsymbol{V}_f \boldsymbol{P}_q$

$$\boldsymbol{Q}_i + \boldsymbol{Q}_i^T = \boldsymbol{E}^T \boldsymbol{B}_i \boldsymbol{E}, \qquad \boldsymbol{B}_i = \boldsymbol{W}_f \operatorname{diag}(\boldsymbol{n}_i)$$

$$\Longrightarrow \int_{\widehat{D}} \frac{\partial P_N u}{\partial x_i} v + \int_{\widehat{D}} u \frac{\partial P_N v}{\partial x_i} = \int_{\partial \widehat{D}} (P_N u)(P_N v)\, \widehat{n}_i.$$

A "decoupled" block SBP operator

- Quadrature may not contain boundary points: complicated interface terms for coupling between neighboring elements or imposing BCs.
- On D^k with unit normal vector $\boldsymbol{n}$: approx. derivative w.r.t x_i.

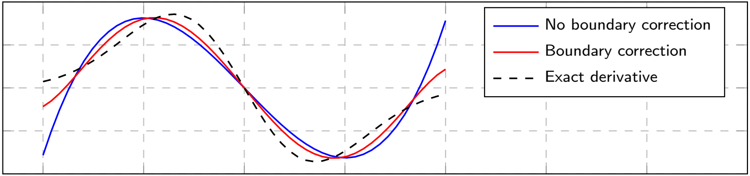
$$\boldsymbol{Q}_N^i = \begin{bmatrix} \boldsymbol{Q}_i - \frac{1}{2}\boldsymbol{E}^T\boldsymbol{B}_i\boldsymbol{E} & \frac{1}{2}\boldsymbol{E}^T\boldsymbol{B}_i \\ -\frac{1}{2}\boldsymbol{B}_i\boldsymbol{E} & \frac{1}{2}\boldsymbol{B}_i \end{bmatrix},$$

- If $\boldsymbol{Q}_i$ satisfies a generalized SBP property, $\boldsymbol{D}_N^i$ satisfies the SBP property

$$\boldsymbol{D}_N^i = \begin{bmatrix} \boldsymbol{W} & \\ & \boldsymbol{W}_f \end{bmatrix}^{-1} \boldsymbol{Q}_N^i, \qquad \boldsymbol{B}_N^i = \begin{bmatrix} \boldsymbol{0} & \\ & \boldsymbol{B}_i \end{bmatrix},$$

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

Decoupled SBP operators add boundary corrections



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

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

- Reduces to traditional SBP operator under appropriate quadrature.
- Equivalent to a skew-symmetric variational problem for $u(\boldsymbol{x}) \approx f\frac{\partial g}{\partial x}$.

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

Talk outline

1 Stability of high order DG: linear vs nonlinear PDEs

2 Summation-by-parts and high order DG

3 Entropy stable formulations and flux differencing

4 Numerical experiments
- Triangular and tetrahedral meshes
- Quadrilateral and hexahedral meshes
- Hybrid and non-conforming meshes

Burgers' equation: decoupled SBP and energy stability

- Revisit split form of Burgers' equation:

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

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

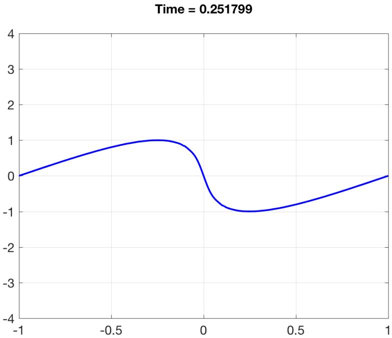
$$\boldsymbol{u} = \begin{bmatrix} \boldsymbol{V}_q \\ \boldsymbol{V}_f \end{bmatrix} \widehat{\boldsymbol{u}}, \qquad \boldsymbol{f}^* = \boldsymbol{f}^*(u^+, u) = \text{numerical flux}$$

$$\boldsymbol{M}\frac{\mathrm{d}\widehat{\boldsymbol{u}}}{\mathrm{dt}} + \frac{1}{3}\begin{bmatrix} \boldsymbol{V}_q \\ \boldsymbol{V}_f \end{bmatrix}^T \left(\boldsymbol{Q}_N\left(\boldsymbol{u}^2\right) + \mathrm{diag}\left(\boldsymbol{u}\right)\boldsymbol{Q}_N\boldsymbol{u}\right) + \boldsymbol{V}_f^T\boldsymbol{B}\boldsymbol{f}^* = 0.$$

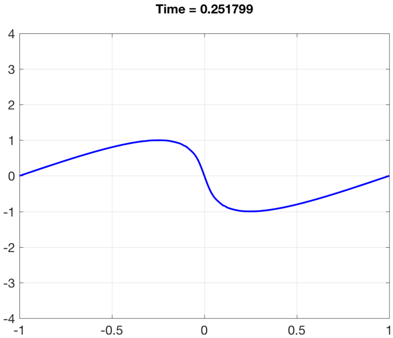
- Formulation is energy stable for arbitrary volume quadratures

$$\frac{\mathrm{d}}{\mathrm{dt}}\widehat{\boldsymbol{u}}^T\boldsymbol{M}\widehat{\boldsymbol{u}} = \frac{\partial}{\partial t}\left\|u_h\right\|^2 \leq 0$$

Burgers' equation: energy stable shock solution

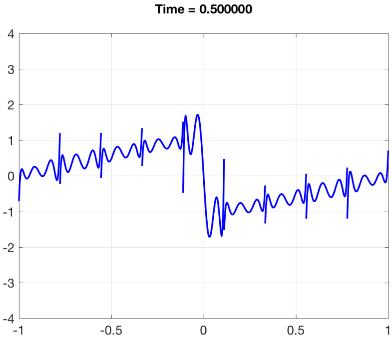


(a) $\tau = 0$

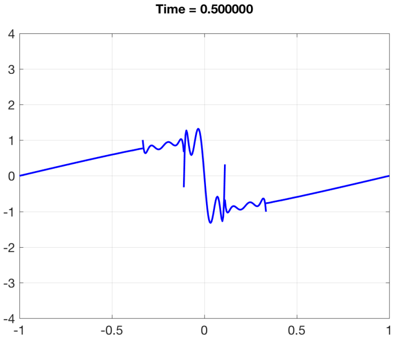


(b) $\tau = 1$

Burgers' equation: energy stable shock solution

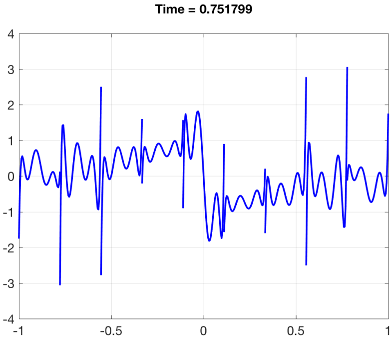


(a) $\tau = 0$

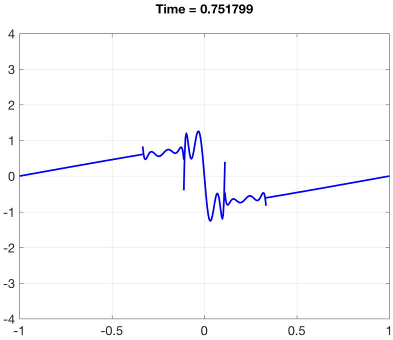


(b) $\tau = 1$

Burgers' equation: energy stable shock solution

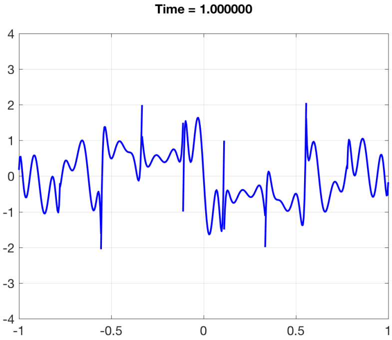


(a) $\tau = 0$

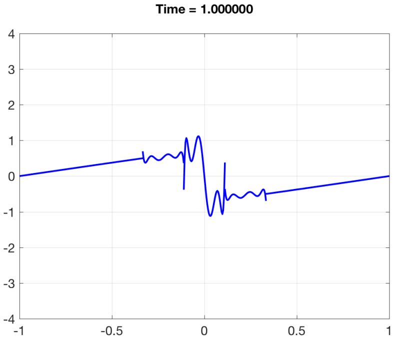


(b) $\tau = 1$

Burgers' equation: energy stable shock solution

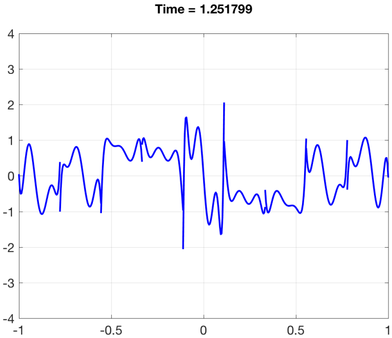


(a) $\tau = 0$

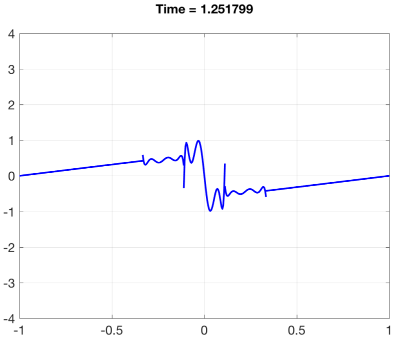


(b) $\tau = 1$

Burgers' equation: energy stable shock solution

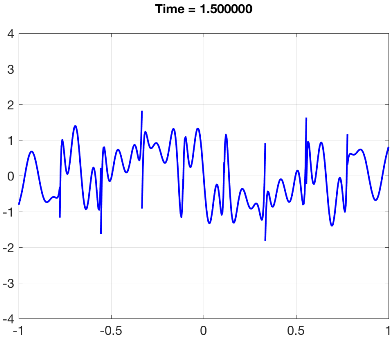


(a) $\tau = 0$

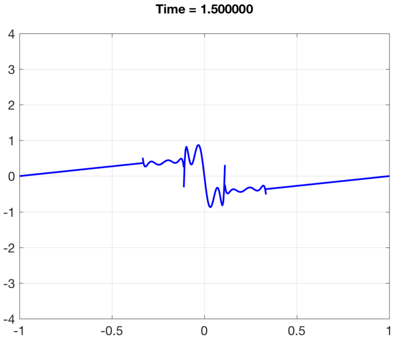


(b) $\tau = 1$

Burgers' equation: energy stable shock solution

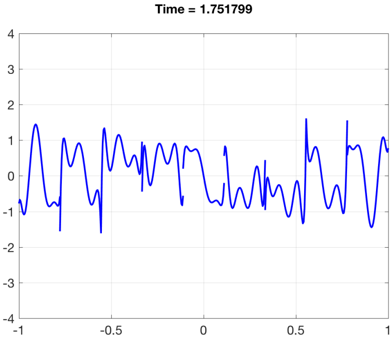


(a) $\tau = 0$

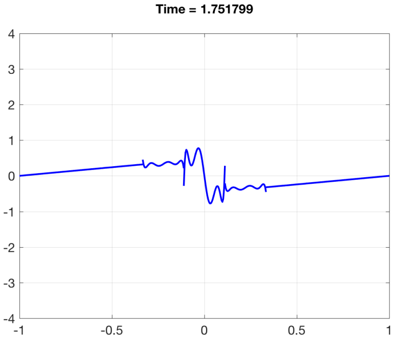


(b) $\tau = 1$

Burgers' equation: energy stable shock solution

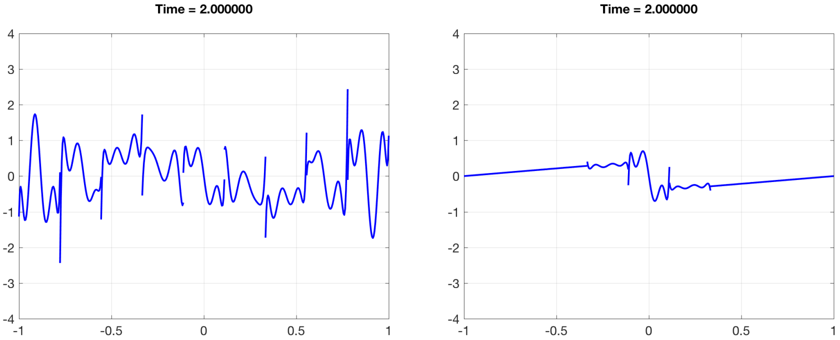


(a) $\tau = 0$



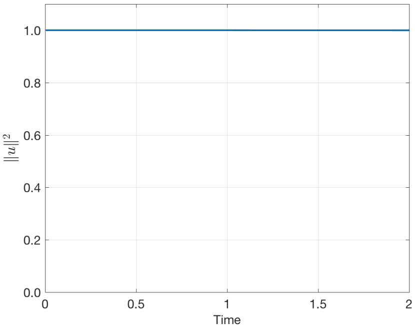
(b) $\tau = 1$

Burgers' equation: energy stable shock solution

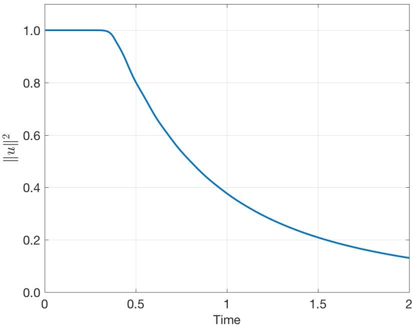


(a) $\tau = 0$ (b) $\tau = 1$

Burgers' equation: energy stable shock solution



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



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

Entropy conservative finite volume fluxes

- Tadmor's entropy conservative 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: use finite volume fluxes to evaluate derivatives.

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

Entropy conservative finite volume fluxes

- Tadmor's entropy conservative 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: use finite volume fluxes to evaluate derivatives.

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

Entropy conservative finite volume fluxes

- Tadmor's entropy conservative 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: use finite volume fluxes to evaluate derivatives.

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

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

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

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 the decoupled SBP operator $\boldsymbol{D}_N$ + polynomial L^2 projection and lifting matrices.

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

- Simpler Hadamard product reformulation: evaluate $\boldsymbol{F}_S$ on-the-fly

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

Flux differencing circumvents the chain rule

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

$$\widetilde{\boldsymbol{v}}^T \left(2\boldsymbol{Q}_N \circ \boldsymbol{F}_S\right)\boldsymbol{1} = \widetilde{\boldsymbol{v}}^T \left(\left(\boldsymbol{B}_N + \boldsymbol{Q}_N - \boldsymbol{Q}_N^T\right) \circ \boldsymbol{F}_S\right)\boldsymbol{1}.$$

- 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 uses $\left(\widetilde{\boldsymbol{v}}_i - \widetilde{\boldsymbol{v}}_j\right)^T \boldsymbol{f}_S\left(\widetilde{\boldsymbol{u}}_i, \widetilde{\boldsymbol{u}}_j\right) = \psi(\widetilde{\boldsymbol{u}}_i) - \psi(\widetilde{\boldsymbol{u}}_j)$: requires entropy variables $\widetilde{\boldsymbol{v}}$ to be a function of conservative variables $\widetilde{\boldsymbol{u}}$.

Flux differencing circumvents the chain rule

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

$$\widetilde{\boldsymbol{v}}^T \left(2\boldsymbol{Q}_N \circ \boldsymbol{F}_S\right)\boldsymbol{1} = \widetilde{\boldsymbol{v}}^T \left(\left(\boldsymbol{B}_N + \boldsymbol{Q}_N - \boldsymbol{Q}_N^T\right) \circ \boldsymbol{F}_S\right)\boldsymbol{1}.$$

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

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

- Proof uses $\left(\widetilde{\boldsymbol{v}}_i - \widetilde{\boldsymbol{v}}_j\right)^T \boldsymbol{f}_S\left(\widetilde{\boldsymbol{u}}_i, \widetilde{\boldsymbol{u}}_j\right) = \psi(\widetilde{\boldsymbol{u}}_i) - \psi(\widetilde{\boldsymbol{u}}_j)$: requires entropy variables $\widetilde{\boldsymbol{v}}$ to be a function of conservative variables $\widetilde{\boldsymbol{u}}$.

Flux differencing circumvents the chain rule

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

$$\widetilde{\boldsymbol{v}}^T \left(2\boldsymbol{Q}_N \circ \boldsymbol{F}_S\right) \boldsymbol{1} = \widetilde{\boldsymbol{v}}^T \left(\left(\boldsymbol{B}_N + \boldsymbol{Q}_N - \boldsymbol{Q}_N^T\right) \circ \boldsymbol{F}_S\right) \boldsymbol{1}.$$

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

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

- Proof uses $\left(\widetilde{\boldsymbol{v}}_i - \widetilde{\boldsymbol{v}}_j\right)^T \boldsymbol{f}_S\left(\widetilde{\boldsymbol{u}}_i, \widetilde{\boldsymbol{u}}_j\right) = \psi(\widetilde{\boldsymbol{u}}_i) - \psi(\widetilde{\boldsymbol{u}}_j)$: requires entropy variables $\widetilde{\boldsymbol{v}}$ to be a function of conservative variables $\widetilde{\boldsymbol{u}}$.

Flux differencing circumvents the chain rule

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

$$\widetilde{\boldsymbol{v}}^T \left(2\boldsymbol{Q}_N \circ \boldsymbol{F}_S\right)\mathbf{1} = \widetilde{\boldsymbol{v}}^T \left(\left(\boldsymbol{B}_N + \boldsymbol{Q}_N - \boldsymbol{Q}_N^T\right) \circ \boldsymbol{F}_S\right)\mathbf{1}.$$

- 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 uses $\left(\widetilde{\boldsymbol{v}}_i - \widetilde{\boldsymbol{v}}_j\right)^T \boldsymbol{f}_S\left(\widetilde{\boldsymbol{u}}_i, \widetilde{\boldsymbol{u}}_j\right) = \psi(\widetilde{\boldsymbol{u}}_i) - \psi(\widetilde{\boldsymbol{u}}_j)$: requires entropy variables $\widetilde{\boldsymbol{v}}$ to be a function of conservative variables $\widetilde{\boldsymbol{u}}$.

Flux differencing circumvents the chain rule

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

$$\widetilde{\boldsymbol{v}}^T \left(2\boldsymbol{Q}_N \circ \boldsymbol{F}_S\right)\mathbf{1} = \widetilde{\boldsymbol{v}}^T \left(\left(\boldsymbol{B}_N + \boldsymbol{Q}_N - \boldsymbol{Q}_N^T\right) \circ \boldsymbol{F}_S\right)\mathbf{1}.$$

- 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 uses $\left(\widetilde{\boldsymbol{v}}_i - \widetilde{\boldsymbol{v}}_j\right)^T \boldsymbol{f}_S\left(\widetilde{\boldsymbol{u}}_i, \widetilde{\boldsymbol{u}}_j\right) = \psi(\widetilde{\boldsymbol{u}}_i) - \psi(\widetilde{\boldsymbol{u}}_j)$: requires entropy variables $\widetilde{\boldsymbol{v}}$ to be a function of 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 $\widetilde{\boldsymbol{u}}$

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

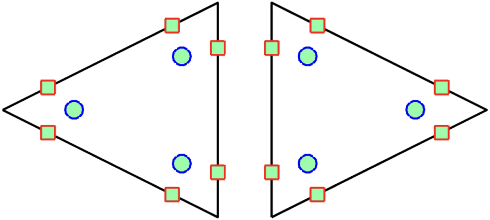
- If $\boldsymbol{v}(\boldsymbol{u})$ is an invertible mapping, this choice of $\widetilde{\boldsymbol{u}}$ ensures that

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

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

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

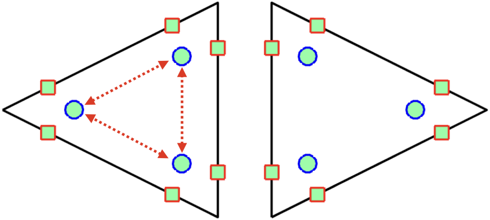
Illustration of main steps of ESDG



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

- 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 and surface nodes.

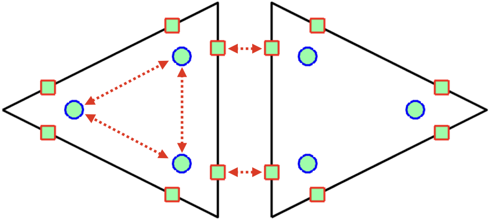
Illustration of main steps of ESDG



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

- 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 and surface nodes.

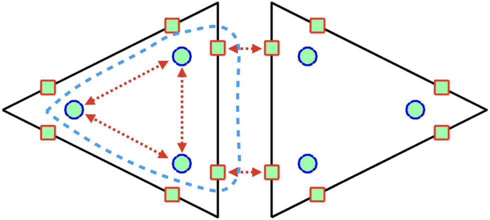
Illustration of main steps of ESDG



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

- 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 and 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 and surface nodes.

A general entropy conservative DG formulation

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*

$$\boldsymbol{M}\frac{\mathrm{d}\widehat{\boldsymbol{u}}}{\mathrm{dt}} + \sum_{i=1}^{d} \begin{bmatrix} \boldsymbol{V}_q \\ \boldsymbol{V}_f \end{bmatrix}^T \left(2\boldsymbol{Q}_N^i \circ \boldsymbol{F}_S^i\right)\boldsymbol{1} + \boldsymbol{V}_f^T \boldsymbol{B}_i \left(\boldsymbol{f}_S^i(\widetilde{\boldsymbol{u}}^+, \widetilde{\boldsymbol{u}}) - \boldsymbol{f}^i(\widetilde{\boldsymbol{u}})\right) = 0.$$

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.*

Talk outline

1 Stability of high order DG: linear vs nonlinear PDEs

2 Summation-by-parts and high order DG

3 Entropy stable formulations and flux differencing

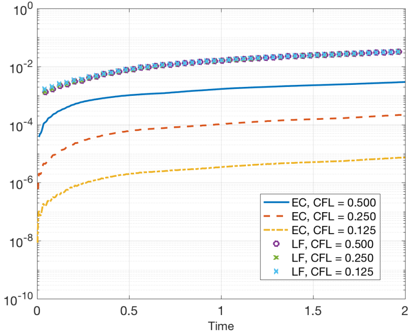
4 Numerical experiments
- Triangular and tetrahedral meshes
- Quadrilateral and hexahedral meshes
- Hybrid and non-conforming meshes

Talk outline

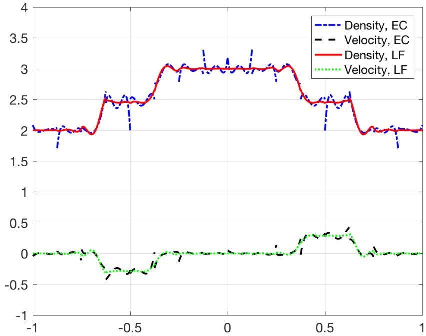
1. Stability of high order DG: linear vs nonlinear PDEs
2. Summation-by-parts and high order DG
3. Entropy stable formulations and flux differencing
4. Numerical experiments
   - Triangular and tetrahedral meshes
   - Quadrilateral and hexahedral meshes
   - Hybrid and non-conforming meshes

Conservation of entropy: semi-discrete vs. fully discrete

$$\Delta S(\boldsymbol{u}) = |S(\boldsymbol{u}(x,t)) - S(\boldsymbol{u}(x,0))| \to 0 \text{ as as } \Delta t \to 0.$$



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

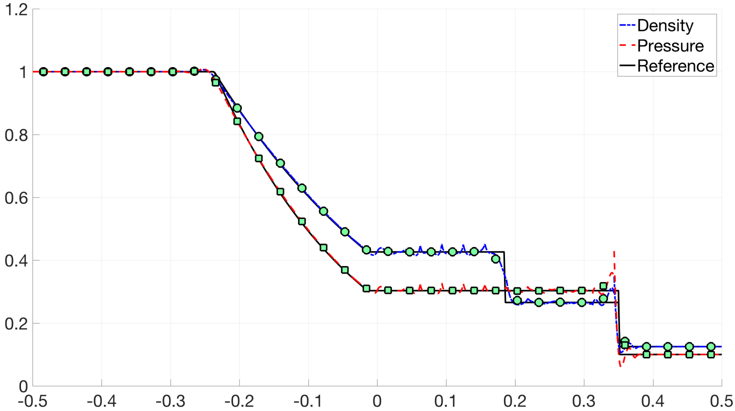


(b) $\rho(x), u(x)$ ($N = 4, K = 16$)

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

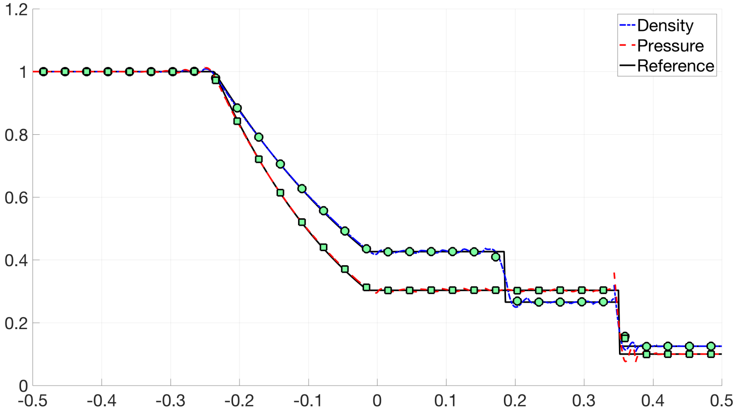
- Circles are cell averages.
- CFL of .125 used for both GLL-$(N+1)$and GQ-$(N+2)$.



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

1D Sod shock tube

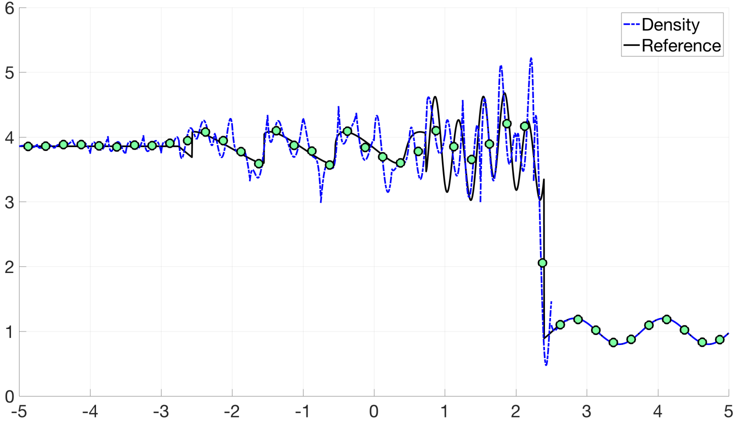
- Circles are cell averages.
- CFL of .125 used for both GLL-$(N+1)$and GQ-$(N+2)$.



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

1D sine-shock interaction

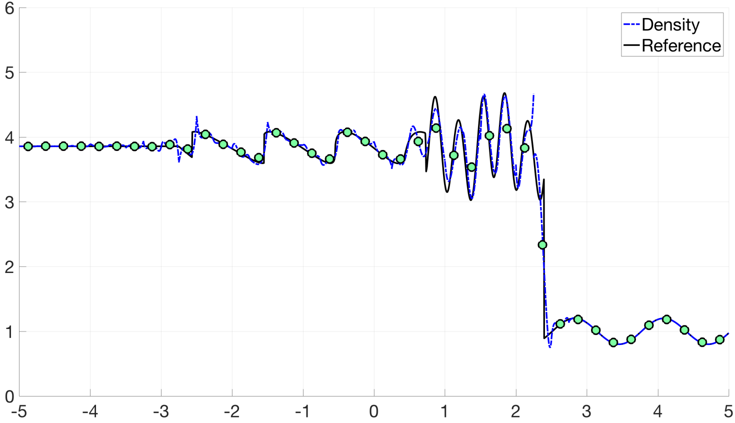
- GQ-($N+2$) does need a smaller CFL (.05 vs .125) for stability.



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

1D sine-shock interaction

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



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

Talk outline

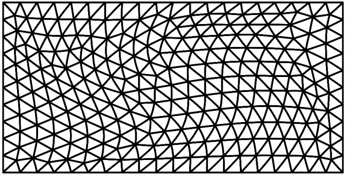
1 Stability of high order DG: linear vs nonlinear PDEs

2 Summation-by-parts and high order DG

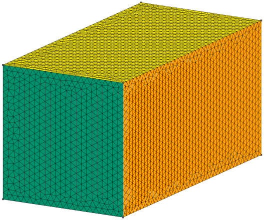
3 Entropy stable formulations and flux differencing

4 Numerical experiments
- Triangular and tetrahedral meshes
- Quadrilateral and hexahedral meshes
- Hybrid and non-conforming meshes

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



(a) 2D triangular mesh



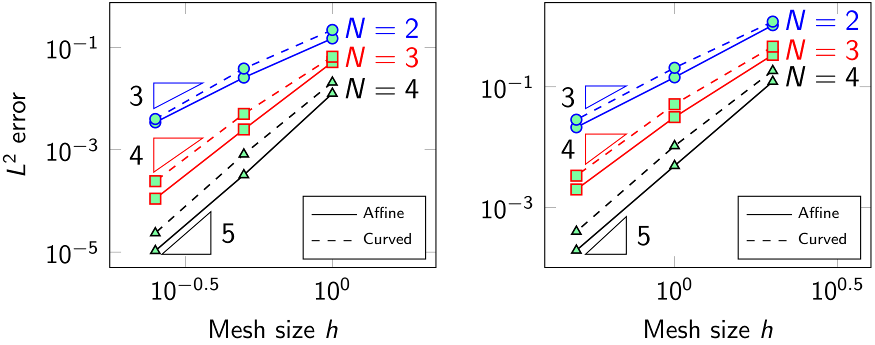
(b) 3D tetrahedral mesh

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

- Entropy stability: needs discrete geometric conservation law (GCL).
- Generalized "weight-adjusted" mass lumping for curved meshes.
- 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.

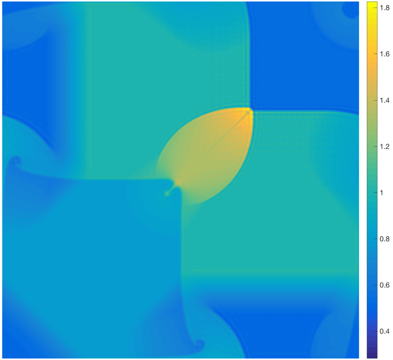
Visbal and Gaitonde (2002). On the Use of Higher-Order Finite-Difference Schemes on Curvilinear and Deforming Meshes.

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.*

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

Inviscid Taylor-Green vortex

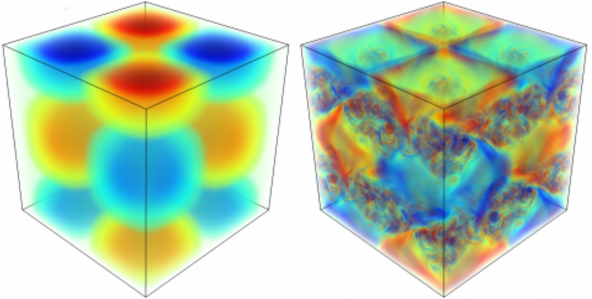


Figure: Isocontours of z-vorticity for Taylor-Green at $t = 0, 10$ seconds.

- 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

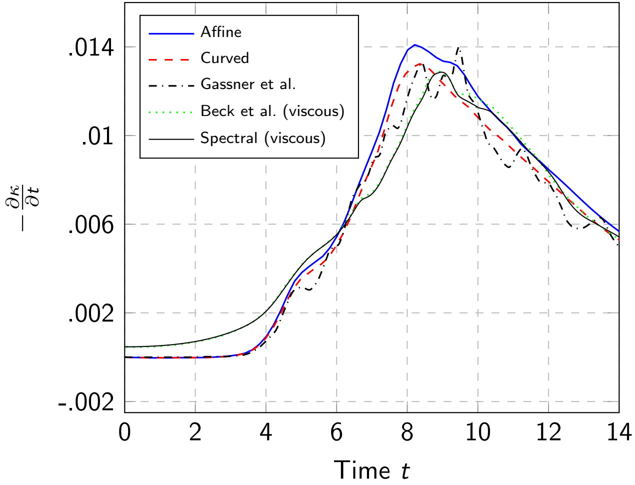


Figure: Evolution of kinetic energy $\kappa(t)$ and kinetic energy dissipation rate $-\frac{\partial \kappa}{\partial t}$ for $N = 3, h = \pi/8, \text{CFL} = .25$ on affine and curved meshes.

Talk outline

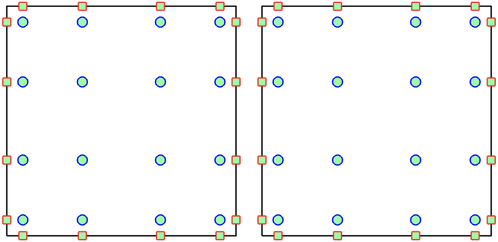
1 Stability of high order DG: linear vs nonlinear PDEs

2 Summation-by-parts and high order DG

3 Entropy stable formulations and flux differencing

4 Numerical experiments
- Triangular and tetrahedral meshes
- Quadrilateral and hexahedral meshes
- Hybrid and non-conforming meshes

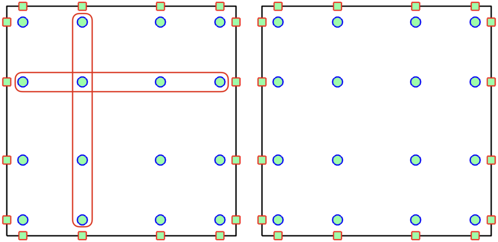
Entropy stable Gauss collocation: main steps



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

- $(N+1)$-point Gauss quadrature reduces to a collocation scheme.
- Advantage over tetrahedral elements: tensor product structure.
- Reduces computational costs from $O(N^6)$ to $O(N^4)$ in 3D.

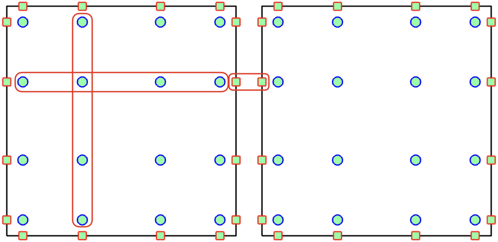
Entropy stable Gauss collocation: main steps



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

- $(N+1)$-point Gauss quadrature reduces to a collocation scheme.
- Advantage over tetrahedral elements: tensor product structure.
- Reduces computational costs from $O(N^6)$ to $O(N^4)$ in 3D.

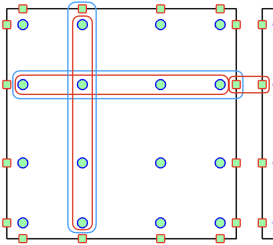
Entropy stable Gauss collocation: main steps



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

- $(N+1)$-point Gauss quadrature reduces to a collocation scheme.
- Advantage over tetrahedral elements: tensor product structure.
- Reduces computational costs from $O(N^6)$ to $O(N^4)$ in 3D.

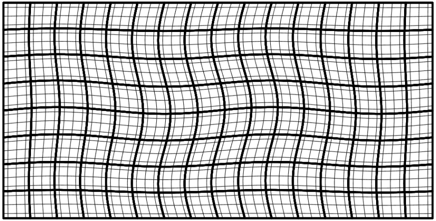
Entropy stable Gauss collocation: main steps



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

- $(N+1)$-point Gauss quadrature reduces to a collocation scheme.
- Advantage over tetrahedral elements: tensor product structure.
- Reduces computational costs from $O(N^6)$ to $O(N^4)$ in 3D.

Gauss quadrature improves errors on curved meshes



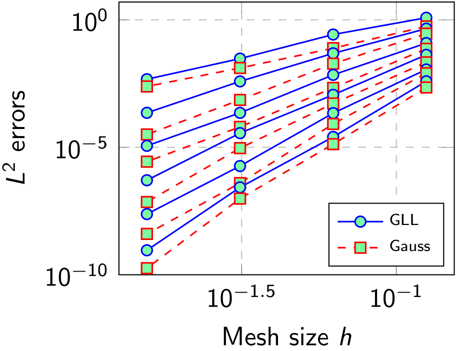
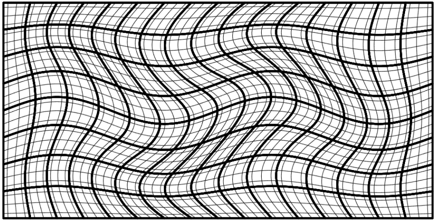


Figure: L^2 errors for the 2D isentropic vortex at time $T = 5$ for degree $N = 2, \ldots, 7$ GLL and Gauss collocation schemes (similar behavior in 3D).

Gauss quadrature improves errors on curved meshes



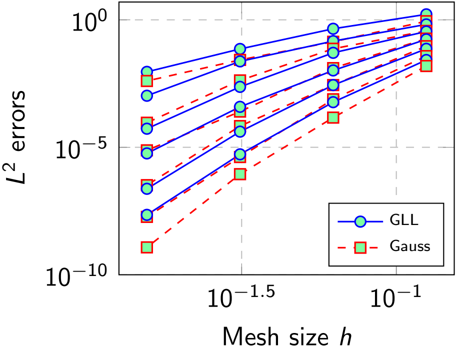
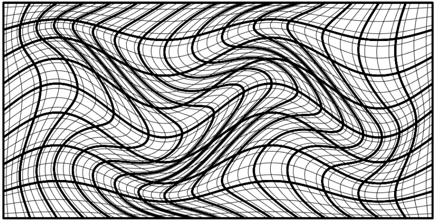


Figure: L^2 errors for the 2D isentropic vortex at time $T = 5$ for degree $N = 2, \ldots, 7$ GLL and Gauss collocation schemes (similar behavior in 3D).

Gauss quadrature improves errors on curved meshes



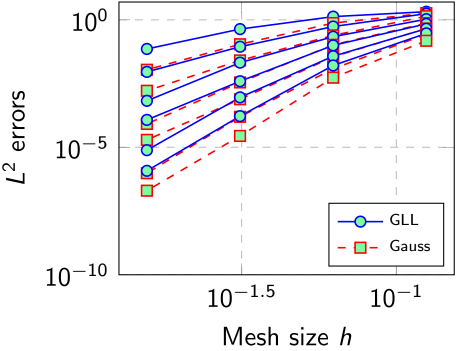
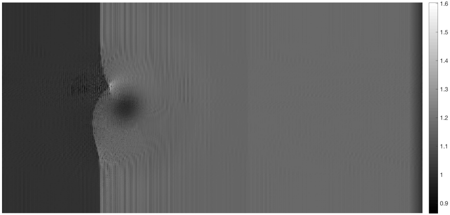
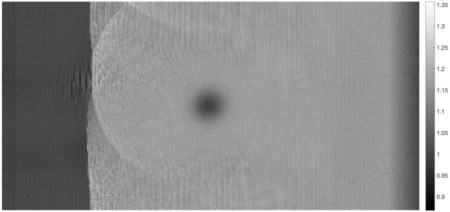


Figure: L^2 errors for the 2D isentropic vortex at time $T = 5$ for degree $N = 2, \ldots, 7$ GLL and Gauss collocation schemes (similar behavior in 3D).

Shock vortex interaction



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

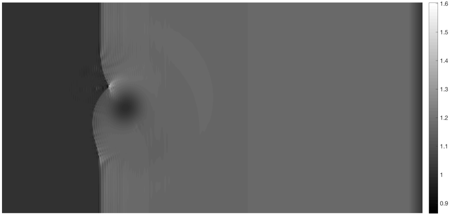


(b) Entropy conservative flux, $T = .7$

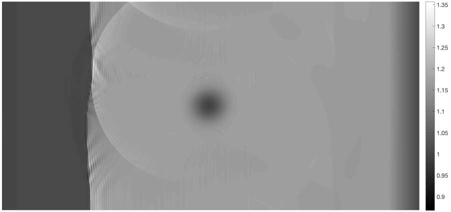
Figure: Shock vortex interaction problem using high order entropy stable Gauss collocation schemes with $N = 4, h = 1/100$.

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, $T = .3$



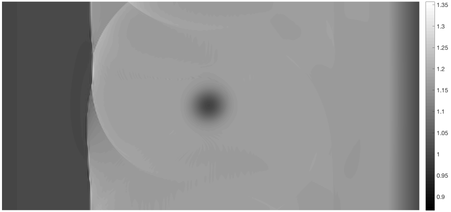
(b) Lax-Friedrichs flux, $T = .7$

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, $T = .3$

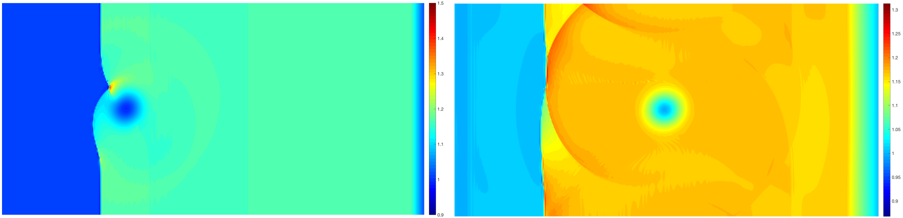


(b) Matrix dissipation flux, $T = .7$

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



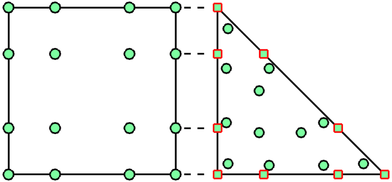
(a) Matrix dissipation flux, $T = .3$ (b) Matrix dissipation flux, $T = .7$

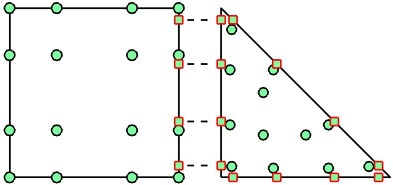
Figure: Shock vortex interaction problem using high order entropy stable Gauss collocation schemes with $N = 4, h = 1/100$.

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

Talk outline

Mixed quadrilateral-triangle meshes

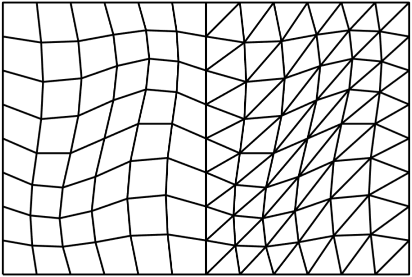

(a) No SBP (tri. under-integrated)


(b) No SBP (quad. under-integrated)

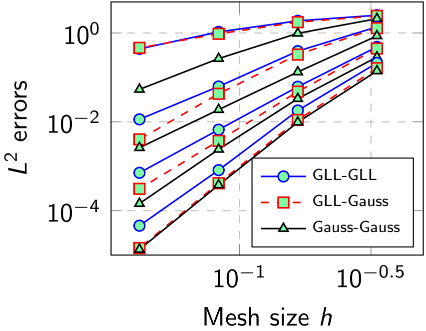
- SBP property requires sufficiently accurate quadrature.
- Skew-symmetric formulation relaxes requirements on quadrature accuracy for entropy stability:

$$\boldsymbol{M}\frac{\mathrm{d}\widehat{\boldsymbol{u}}}{\mathrm{d}t} + \sum_{i=1}^{d} \begin{bmatrix} \boldsymbol{V}_q \\ \boldsymbol{V}_f \end{bmatrix}^T \left(\left(\boldsymbol{Q}_N^i - \left(\boldsymbol{Q}_N^i\right)^T\right) \circ \boldsymbol{F}_S^i\right)\boldsymbol{1} + \boldsymbol{V}_f^T \boldsymbol{B}_i \boldsymbol{f}_S^i(\widetilde{\boldsymbol{u}}^+, \widetilde{\boldsymbol{u}}) = 0.$$

Numerical results: mixed triangle-quadrilateral meshes



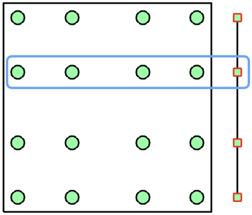
(a) Coarse hybrid mesh



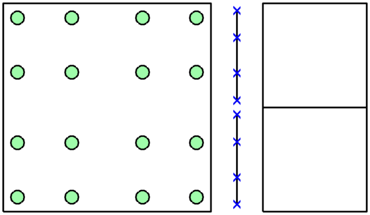
(b) Convergence for $N = 1, 2, 3, 4$

The skew-symmetric formulation guarantees entropy stability for all combinations of GLL and Gauss volume and surface quadratures.

Meshes with non-conforming interfaces



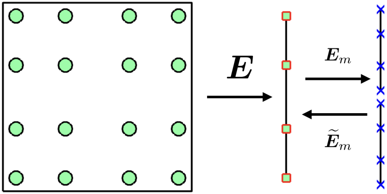
(a) Conforming surface nodes



(b) Non-conforming surface nodes

- Volume/surface nodes interact through $\boldsymbol{f}_S(\boldsymbol{u}_i, \boldsymbol{u}_j)$ and interpolation.
- Weakly couple volume nodes to non-conforming surface nodes by adding conforming "mortar" (via additional blocks in $\boldsymbol{Q}_N$).
- Can reformulate as an entropy stable correction to standard mortar.

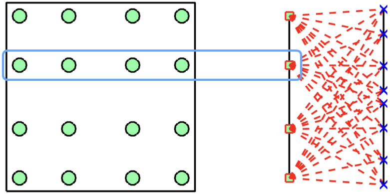
Meshes with non-conforming interfaces



$$\underbrace{\begin{bmatrix} \boldsymbol{Q}_i - \boldsymbol{Q}_i^T & \boldsymbol{E}^T\boldsymbol{B}_i & \\ -\boldsymbol{B}_i\boldsymbol{E} & & \boldsymbol{B}_i\widetilde{\boldsymbol{E}}_m \\ & -\widetilde{\boldsymbol{B}}_i\boldsymbol{E}_m & \end{bmatrix}}_{\text{modified skew operator } \boldsymbol{Q}_N^i - \left(\boldsymbol{Q}_N^i\right)^T}$$

- Volume/surface nodes interact through $\boldsymbol{f}_S(\boldsymbol{u}_i, \boldsymbol{u}_j)$ and interpolation.
- Weakly couple volume nodes to non-conforming surface nodes by adding conforming "mortar" (via additional blocks in $\boldsymbol{Q}_N$).
- Can reformulate as an entropy stable correction to standard mortar.

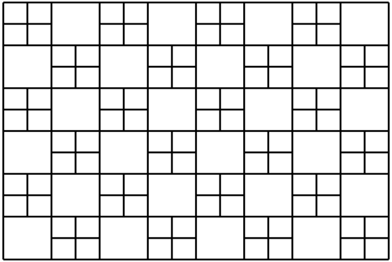
Meshes with non-conforming interfaces



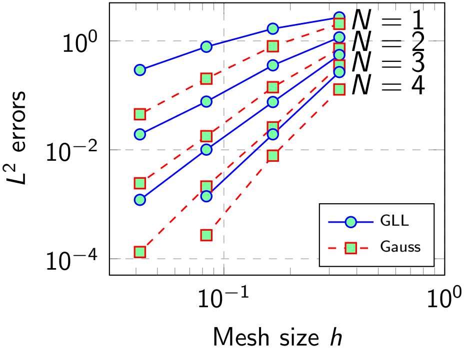
$$\underbrace{\begin{bmatrix} \boldsymbol{Q}_i - \boldsymbol{Q}_i^T & \boldsymbol{E}^T\boldsymbol{B}_i & \\ -\boldsymbol{B}_i\boldsymbol{E} & & \boldsymbol{B}_i\widetilde{\boldsymbol{E}}_m \\ & -\widetilde{\boldsymbol{B}}_i\boldsymbol{E}_m & \end{bmatrix}}_{\text{modified skew operator } \boldsymbol{Q}_N^i - \left(\boldsymbol{Q}_N^i\right)^T}$$

- Volume/surface nodes interact through $\boldsymbol{f}_S(\boldsymbol{u}_i, \boldsymbol{u}_j)$ and interpolation.
- Weakly couple volume nodes to non-conforming surface nodes by adding conforming "mortar" (via additional blocks in $\boldsymbol{Q}_N$).
- Can reformulate as an entropy stable correction to standard mortar.

Numerical results: non-conforming meshes



(a) Coarse non-conforming mesh



(b) Sub-optimal rates if under-integrated

The skew-symmetric formulation guarantees entropy stability for both GLL and Gauss quadratures, but Gauss is more accurate.

Summary and future work

- Entropy stable high order discontinuous Galerkin methods: semi-discrete stability, improved robustness.
- Additional work required for strong shocks, positivity preservation.
- Current work: hybrid and 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.*

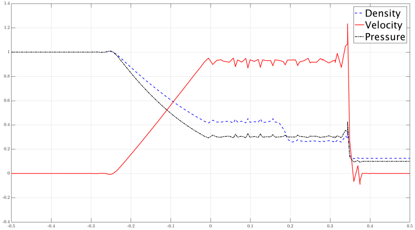
Chan, Wilcox (2018). *On discretely entropy stable weight-adjusted DG methods: curvilinear meshes.*

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

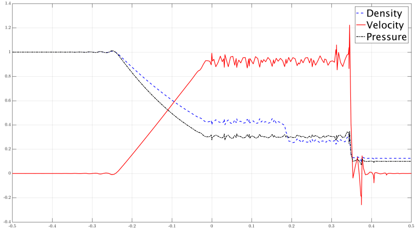
Chan (2017). *On discretely entropy conservative and entropy stable discontinuous Galerkin methods.*

Additional slides

Over-integration is ineffective without L^2 projection



(a) $(N+1)$ points

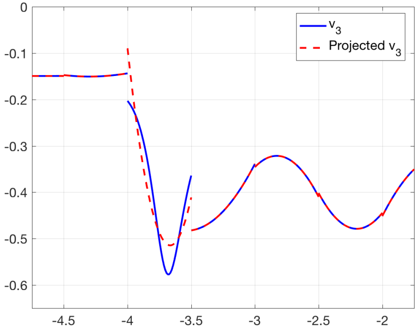


(b) $(N+4)$ points

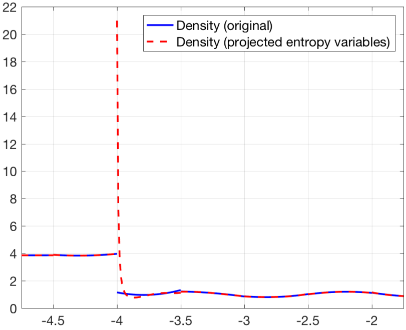
Figure: Numerical results for the Sod shock tube for $N = 4$ and $K = 32$ elements. Over-integrating by increasing the number of quadrature points does not improve solution quality.

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

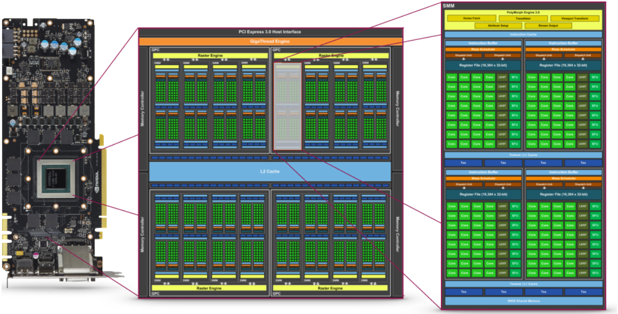


Figure: NVIDIA Maxwell GM204 GPU: 16 cores, 4 SIMD clusters of 32 units.

- 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

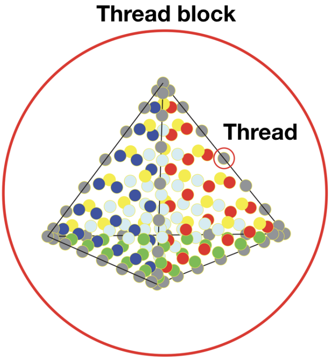


Figure: Thread blocks process elements, threads process degrees of freedom.

- 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

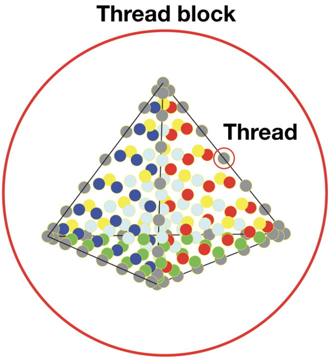


Figure: Thread blocks process elements, threads process degrees of freedom.

- 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.

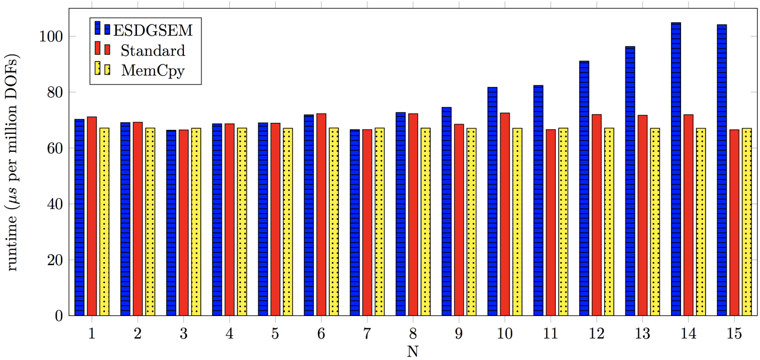
Implementing high order entropy stable DG on GPUs

- "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.*