

Efficient time-domain DG methods for wave propagation

Jesse Chan

¹Department of Computational and Applied Mathematics
Rice University

ICES, UT Austin
April 6, 2017

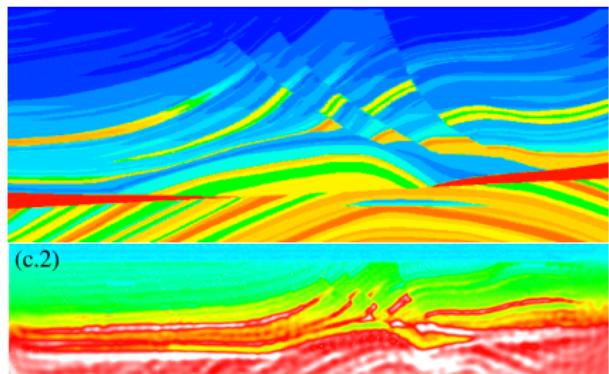
Collaborators and contributors

- T. Warburton (Virginia Tech)
- Russell J. Hewett (TOTAL E&P Research and Technology USA)
- John Evans (U.C. Boulder)

Numerical simulation of wave propagation

Many procedures require **accurately** and **efficiently** solving hyperbolic partial differential equations (PDEs) in realistic settings.

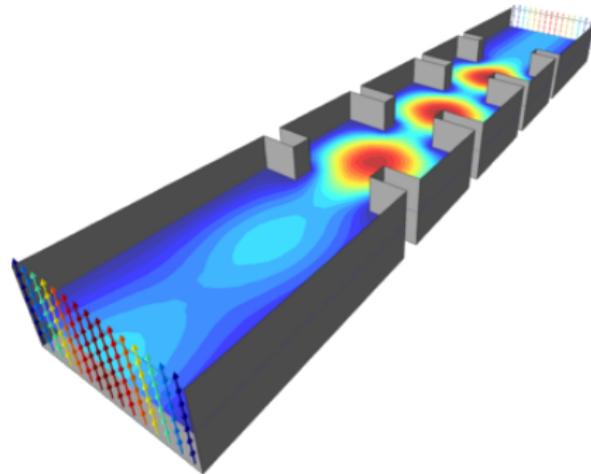
- Seismic and medical imaging
- Engineering design
- Computational fluids



Numerical simulation of wave propagation

Many procedures require **accurately** and **efficiently** solving hyperbolic partial differential equations (PDEs) in realistic settings.

- Seismic and medical imaging
- Engineering design
- Computational fluids

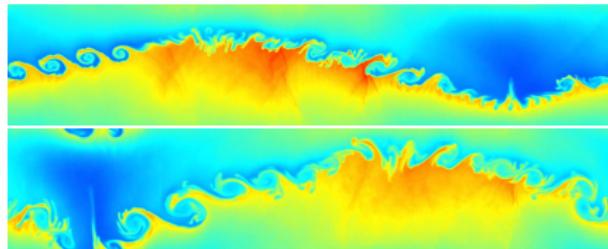


<https://www.comsol.com/model/image/12737/big.png>

Numerical simulation of wave propagation

Many procedures require **accurately** and **efficiently** solving hyperbolic partial differential equations (PDEs) in realistic settings.

- Seismic and medical imaging
- Engineering design
- Computational fluids



High order methods for wave problems

- Accurately represent acoustic and elastic waves.
- Superior performance vs low order methods for equivalent resolution.
- Low numerical dissipation and dispersion errors.
- Lower error per unknown.

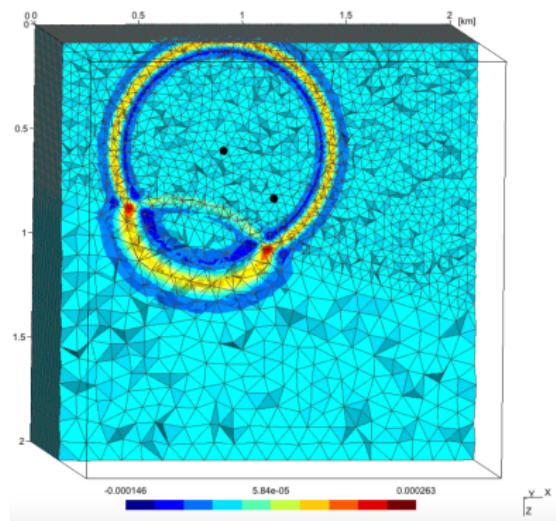
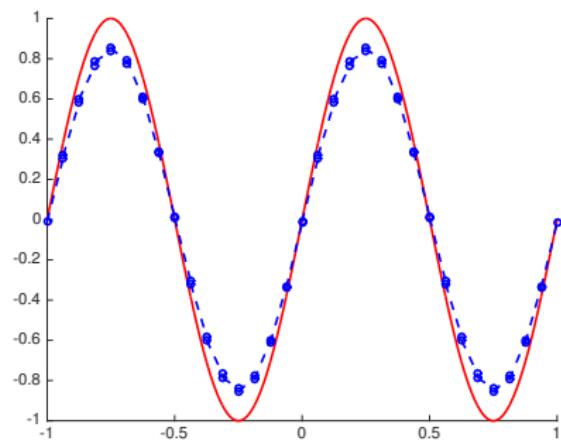


Figure courtesy of Axel Modave.

High order methods for wave problems

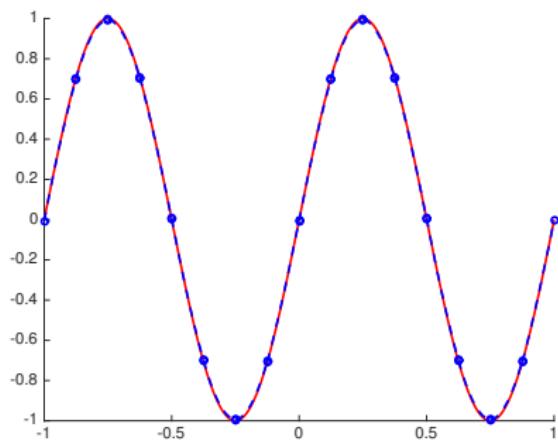
- Accurately represent acoustic and elastic waves.
- Superior performance vs low order methods for equivalent resolution.
- Low numerical dissipation and dispersion errors.
- Lower error per unknown.



Fine linear approximation.

High order methods for wave problems

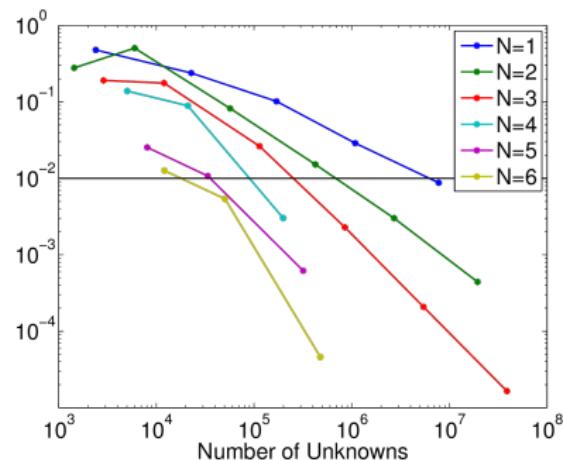
- Accurately represent acoustic and elastic waves.
- Superior performance vs low order methods for equivalent resolution.
- Low numerical dissipation and dispersion errors.
- Lower error per unknown.



Coarse quadratic approximation.

High order methods for wave problems

- Accurately represent acoustic and elastic waves.
- Superior performance vs low order methods for equivalent resolution.
- Low numerical dissipation and dispersion errors.
- Lower error per unknown.

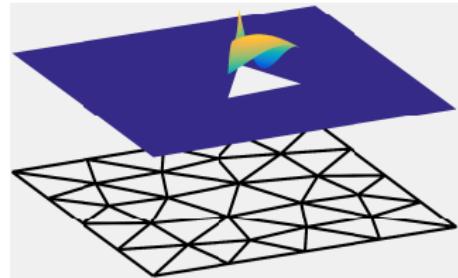


Max errors vs. d.o.f.s.

Discontinuous Galerkin methods

Discontinuous Galerkin (DG) methods:

- Piecewise polynomial approximation.
- Weak continuity across faces.
- Continuous PDE (example: advection)



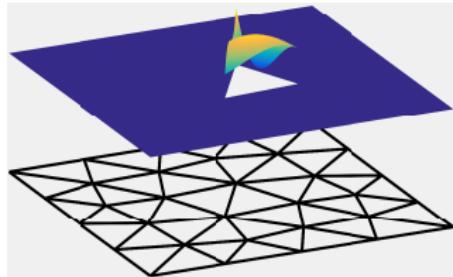
- DG local weak form over D_k with numerical flux \mathbf{f}^* .

$$\int_{D_k} \frac{\partial u}{\partial t} \phi = \int_{D_k} \frac{\partial u}{\partial x} \phi + \int_{\partial D_k} \mathbf{n} \cdot (\mathbf{f}^* - \mathbf{f}(u)) \phi, \quad u, \phi \in V_h$$

Discontinuous Galerkin methods

Discontinuous Galerkin (DG) methods:

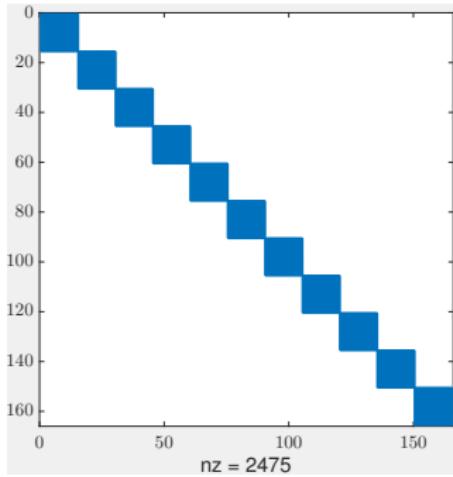
- Piecewise polynomial approximation.
- Weak continuity across faces.



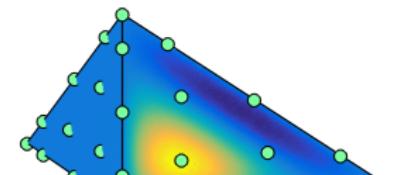
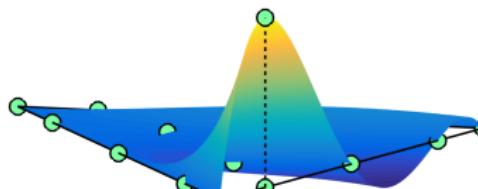
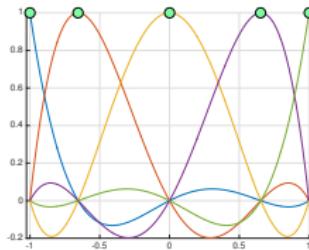
DG yields system of ODEs

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

DG mass matrix decouples across elements,
inter-element coupling only through **A**.



High order nodal discontinuous Galerkin methods



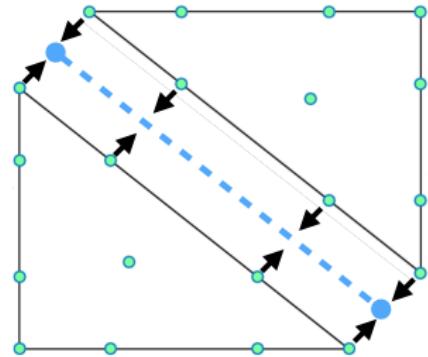
Lagrange (nodal) bases on a line, triangle, tetrahedron.

- Nodal bases defined implicitly through an orthogonal basis.
- Point locations optimized for interpolation and numerical stability.
- Assume **affine** tetrahedra, coefficients **constant** on each element.

Time-domain nodal DG methods

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

- Compute numerical flux at face nodes (**non-local**).
- Compute RHS of (**local**) ODE.
- Evolve (**local**) solution using explicit time integration (RK, AB, etc).



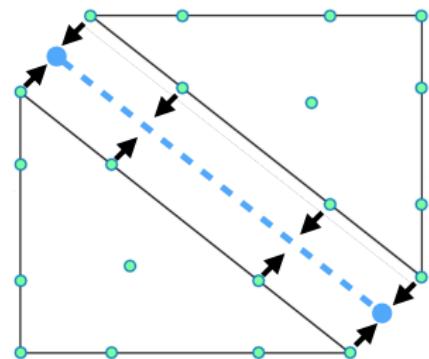
$$\frac{d\mathbf{u}}{dt} = \mathbf{D}_x \mathbf{u} + \sum_{\text{faces}} \mathbf{L}_f (\text{flux}), \quad \mathbf{L}_f = \mathbf{M}^{-1} \mathbf{M}_f.$$

Parallelizable down to individual degrees of freedom!

Time-domain nodal DG methods

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

- Compute numerical flux at face nodes (**non-local**).
- Compute RHS of (**local**) ODE.
- Evolve (**local**) solution using explicit time integration (RK, AB, etc).



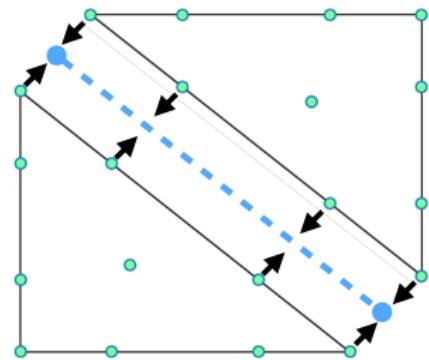
$$\frac{d\mathbf{u}}{dt} = \underbrace{\mathbf{D}_x \mathbf{u}}_{\text{Volume kernel}} + \underbrace{\sum_{\text{faces}} \mathbf{L}_f (\text{flux})}_{\text{Surface kernel}}, \quad \mathbf{L}_f = \mathbf{M}^{-1} \mathbf{M}_f.$$

Parallelizable down to individual degrees of freedom!

Time-domain nodal DG methods

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

- Compute numerical flux at face nodes (**non-local**).
- Compute RHS of (**local**) ODE.
- Evolve (**local**) solution using explicit time integration (RK, AB, etc).



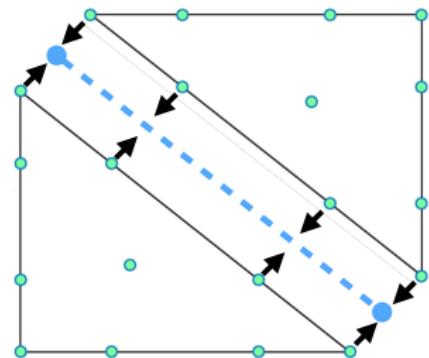
$$\underbrace{\frac{du}{dt}}_{\text{Update kernel}} = \underbrace{D_x u}_{\text{Volume kernel}} + \underbrace{\sum_{\text{faces}} L_f}_{\text{Surface kernel}} (\text{flux}), \quad L_f = M^{-1} M_f.$$

Parallelizable down to individual degrees of freedom!

Time-domain nodal DG methods

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

- Compute numerical flux at face nodes (**non-local**).
- Compute RHS of (**local**) ODE.
- Evolve (**local**) solution using explicit time integration (RK, AB, etc).



$$\underbrace{\frac{du}{dt}}_{\text{Update kernel}} = \underbrace{D_x u}_{\text{Volume kernel}} + \underbrace{\sum_{\text{faces}} L_f}_{\text{Surface kernel}} (\text{flux}), \quad L_f = M^{-1} M_f.$$

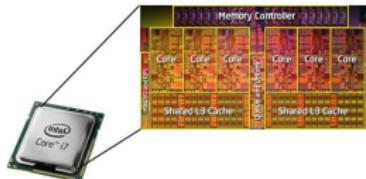
Parallelizable down to individual degrees of freedom!

Computing with Graphics Processing Units (GPUs)

- Scalable but *expensive*: hours for 2D, **days** for 3D with ≈ 100 CPUs.
- Explicit methods: can replace a small cluster with single GPU.
- Highly energy efficient compared to traditional supercomputers.
*... to achieve a [next generation] supercomputer by simply [scaling up] ... you'd need a good-size **nuclear power plant** next door.*
(Kogge 2011)
- GPUs reflect broader many-core trends in computing.



(a) Xeon Phi



(b) Intel i7 CPU

DG maps well to many-core (GPU) architectures

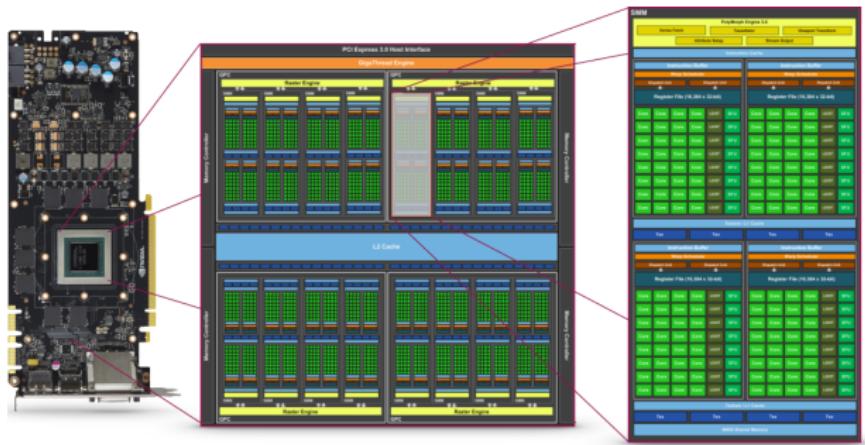


Figure: NVIDIA Maxwell GM204 GPU: 16 cores, 4 SIMD clusters of 32 units.

- Thousands of processing elements organized in synchronized groups.
- Stricter memory constraints (data transfer, limited storage), but ...
- For DG, reduces computational time from **days to hours**.

DG maps well to many-core (GPU) architectures

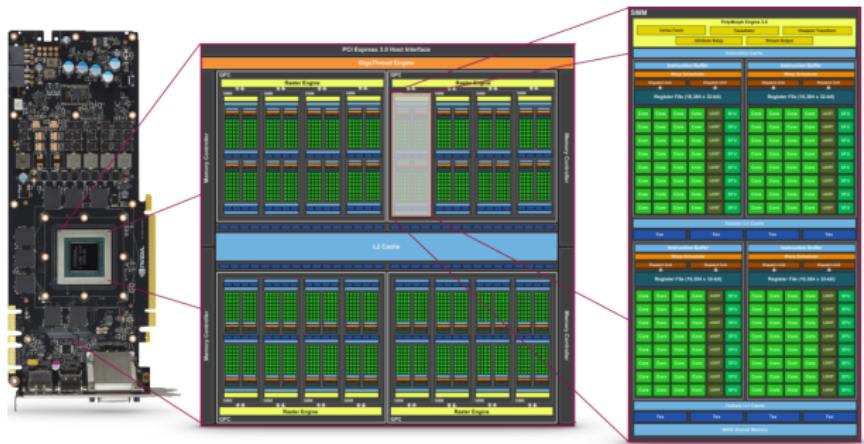
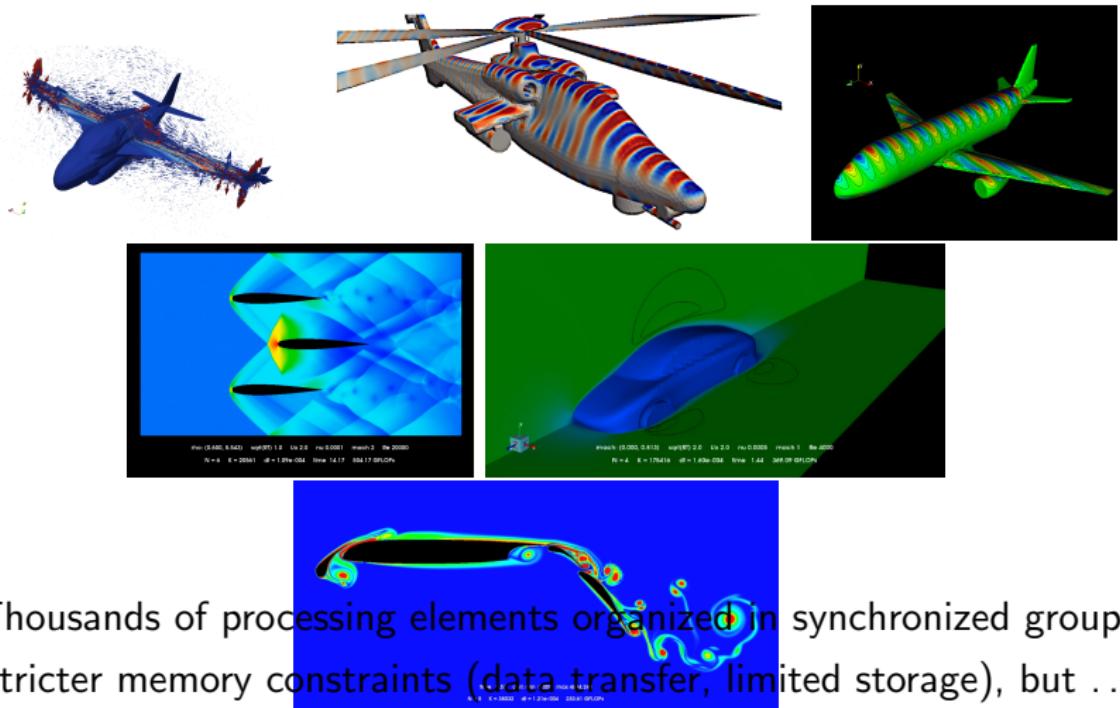


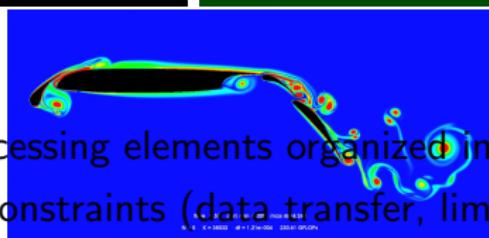
Figure: NVIDIA Maxwell GM204 GPU: 16 cores, 4 SIMD clusters of 32 units.

- Thousands of processing elements organized in synchronized groups.
- Stricter memory constraints (data transfer, limited storage), but ...
- For DG, reduces computational time from **days to hours**.

DG maps well to many-core (GPU) architectures



- Thousands of processing elements organized in synchronized groups.
- Stricter memory constraints (data transfer, limited storage), but ...
- For DG, reduces computational time from **days** to **hours**.



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

Outline: improving efficiency of time-domain DG

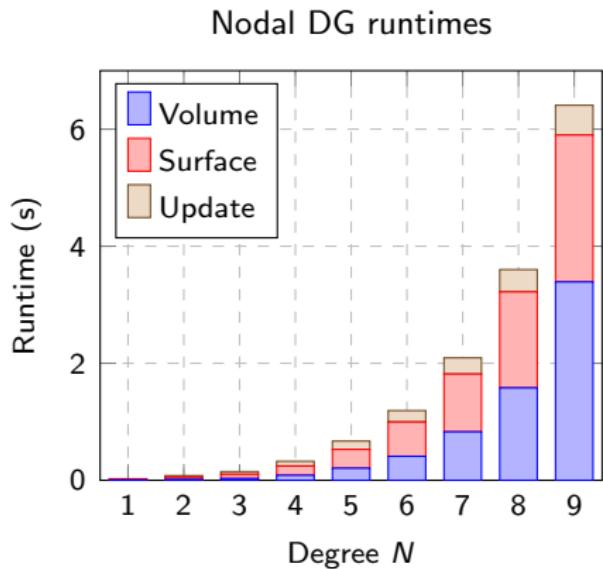
- Optimize existing DG methods on GPUs.
- Address simplifying assumptions (constant coefficients, affine tets).

Outline: improving efficiency of time-domain DG

- Optimize existing DG methods on GPUs.
- Address simplifying assumptions (constant coefficients, affine tets).

Computational costs at high orders of approximation

Problem: (tetrahedral) DG at high orders becomes **very** expensive!



- Large **dense** matrices: $O(N^6)$ work per tet.
- Very high orders usually use tensor-product elements.
- $O(N^4)$ vs $O(N^6)$ cost, but less geometric flexibility.

DG runtimes for 50 timesteps, 98304 elements.

Spectral element methods

- Tensor product elements, Gauss-Legendre-Lobatto nodal basis.
- $O(N^{d+1})$ vs $O(N^{2d})$ work per element (computing derivatives).
- Hexahedral mesh generation more difficult.

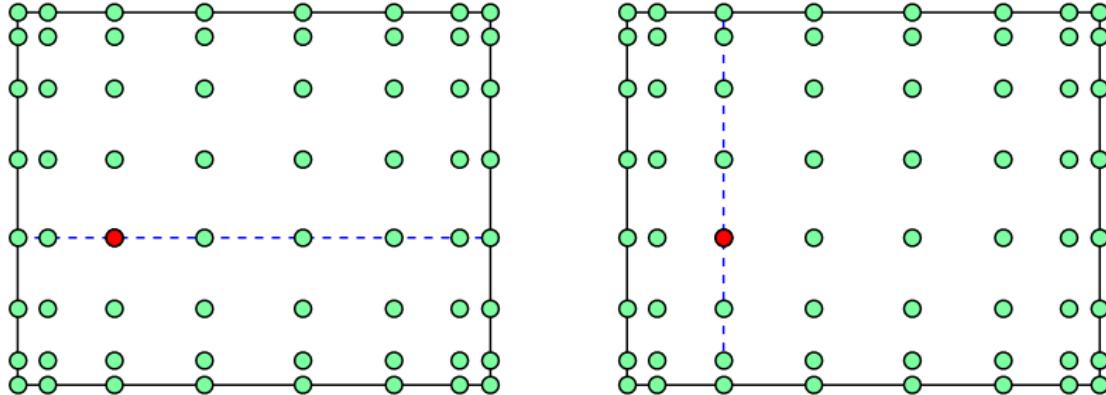


Figure: Spectral element stencils for $N = 7$ (orders $N > 10$ not uncommon!).

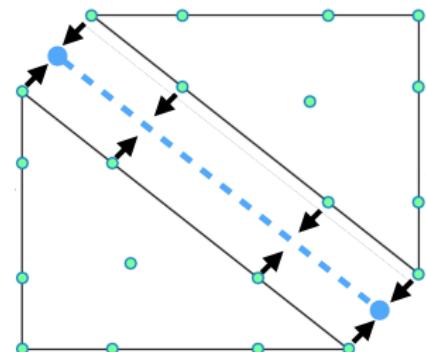
Fischer, Ronquist 1994. Spectral element methods for large scale parallel Navier-Stokes calculations.

Shepherd and Johnson 2008. Hexahedral mesh generation constraints.

High order nodal DG on tetrahedral meshes

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

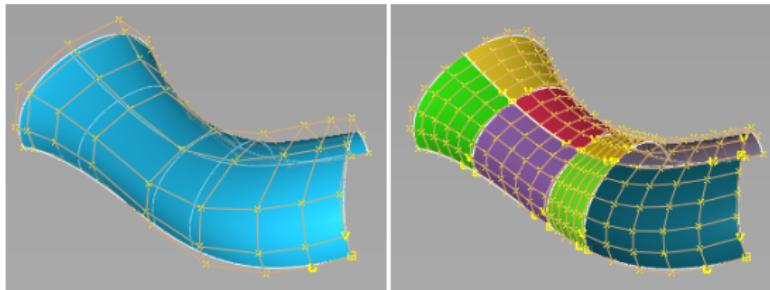
- Nodal bases: reduce the cost of computing numerical fluxes.
- No special structure in nodal derivative/lift matrices.
- $O(N^3)$ unknowns in 3D; $O(N^6)$ costs for applying **dense** matrices.



Derivative and lift matrices depend on the basis:
can we choose one that is efficient (and numerically stable)?

Bernstein-Bezier bases for finite element methods

- Geometry, graphics, Computer Aided Design (CAD).



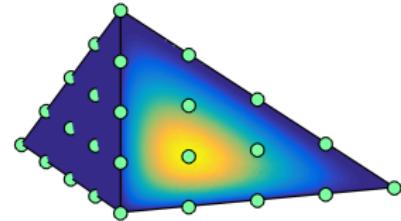
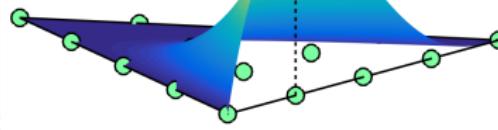
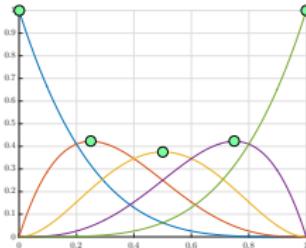
- Recent developments: optimal complexity algorithms for quadrature-based integration, assembly of finite element matrices.
- Is Bernstein-Bezier useful for quadrature-free DG methods?

Split multi-span NURBS surfaces into Bezier patches, <https://knowledge.autodesk.com>

Ainsworth et al. 2011. Bernstein-Bezier finite elements of arbitrary order and optimal assembly procedures.

Kirby 2011. Fast simplicial finite element algorithms using Bernstein polynomials.

Bernstein-Bezier polynomial bases on simplices



Each function attains its maximum at an equispaced lattice point of a d -simplex.

- Simple expression in 1D

$$B_i^N(x) = x^i(1-x)^{N-i}, \quad 0 \leq x \leq 1.$$

- Barycentric monomials on a d -simplex. For a tetrahedron,

$$B_{ijkl}^N(\lambda_0, \lambda_1, \lambda_2, \lambda_3) = \frac{N!}{i!j!k!l!} \lambda_0^i \lambda_1^j \lambda_2^k \lambda_3^l, \quad i + j + k + l = N.$$

- Similar structure to nodal basis (vertex, edge, face, interior functions).

Bernstein-Bezier derivatives and degree elevation in 1D

- Simple differentiation of Bernstein polynomials

$$\frac{\partial B_i^N(x)}{\partial x} = N \left(B_{i-1}^{N-1}(x) - B_i^{N-1}(x) \right).$$

- Simple degree elevation of Bernstein polynomials

$$B_i^{N-1}(x) = \left(\frac{N-i}{N} \right) B_i^N(x) - \left(\frac{i+1}{N} \right) B_{i+1}^N(x).$$

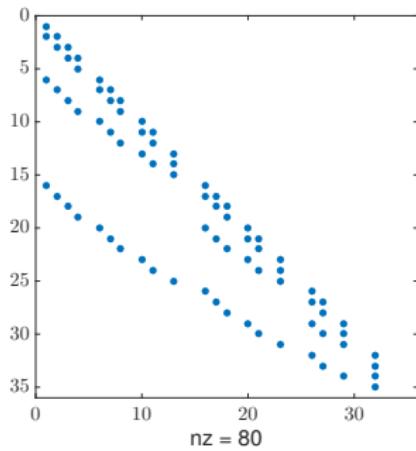
- Combine to get expansion of Bernstein derivatives

$$\frac{\partial B_i^N(x)}{\partial x} = a_i^N B_{i-1}^N(x) + b_i^N B_i^N(x) - c_i^N B_{i+1}^N(x).$$

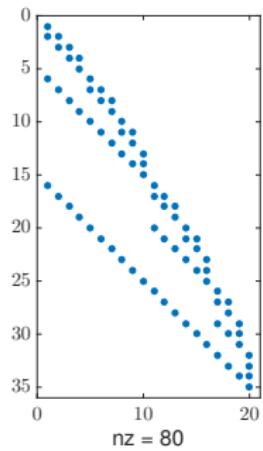
Implies 1D derivative matrix \mathbf{D}_x is **sparse** (tridiagonal).

Bernstein-Bezier derivative and degree elevation in 3D

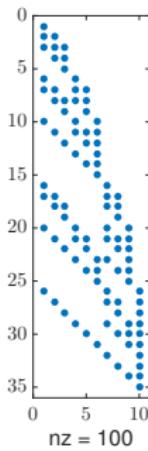
- Bernstein-Bezier barycentric differentiation matrices very sparse.
- Degree elevation matrices \mathbf{E}_{N-i}^N are sparse (for consecutive degrees).
- Higher degree elevation \rightarrow product of matrices $\mathbf{E}_{N-2}^N = \mathbf{E}_{N-1}^N \mathbf{E}_{N-2}^{N-1}$.



(a) Derivative matrix w.r.t.
first barycentric coordinate.



(b) Deg. elevation
matrix \mathbf{E}_{N-1}^N



(c) \mathbf{E}_{N-2}^N

Stencils for Bernstein-Bezier derivative matrices

- Stencil sizes at most $(d + 1)$ in d dimensions.
- Compute derivatives w.r.t. barycentric coordinates.
- Stencil values are identical for each barycentric coordinate.

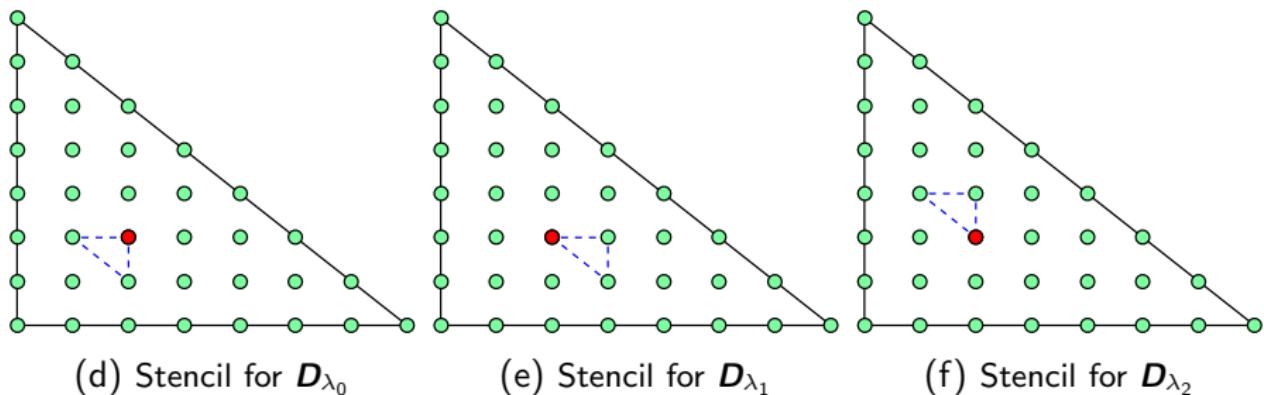
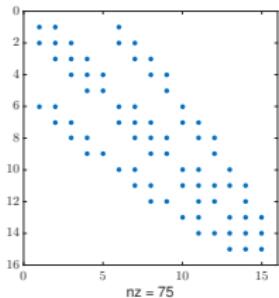
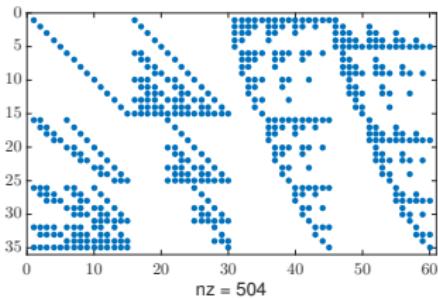
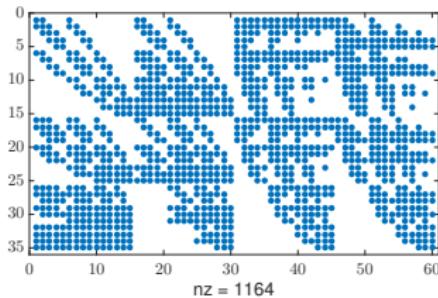


Figure: Bernstein-Bezier stencils for a single node (in red) $N = 7$.

Factorization of the Bernstein lift operator

The Bernstein-Bezier lift matrix \mathbf{L} admits a factorization of the form

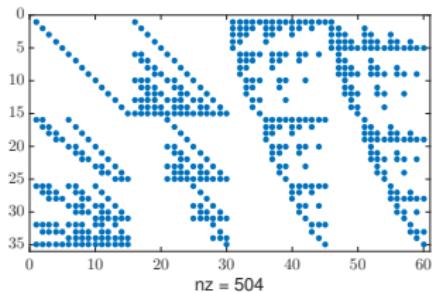
$$\mathbf{L} = \mathbf{E}_L \begin{pmatrix} \mathbf{L}_0 & & & \\ & \mathbf{L}_0 & & \\ & & \mathbf{L}_0 & \\ & & & \mathbf{L}_0 \end{pmatrix}.$$



Factorization of the Bernstein lift operator

The Bernstein-Bezier lift matrix \mathbf{L} admits a factorization of the form

$$\mathbf{L} = \mathbf{E}_L \begin{pmatrix} \mathbf{L}_0 & & & \\ & \mathbf{L}_0 & & \\ & & \mathbf{L}_0 & \\ & & & \mathbf{L}_0 \end{pmatrix}.$$



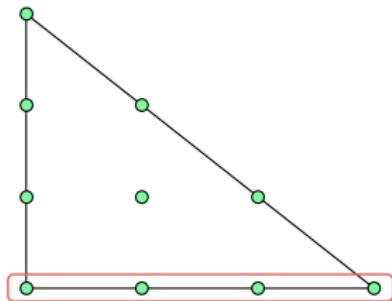
$$\mathbf{E}_L^1 = \begin{bmatrix} \mathbf{I} \\ \ell_1 (\mathbf{E}_{N-1}^N)^T \\ \vdots \\ \ell_N (\mathbf{E}_0^N)^T \end{bmatrix}.$$

$$\mathbf{E}_L = [\mathbf{E}_L^1 \mid \dots \mid \mathbf{E}_L^4] \text{ (4 faces).}$$

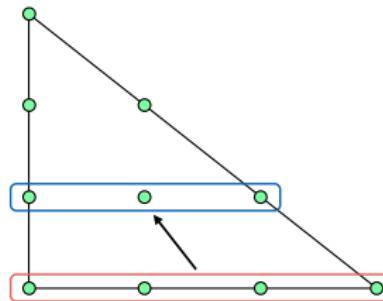
2D degree reduction matrices $(\mathbf{E}_i^N)^T$.

Bernstein-Bezier lift matrix: optimal complexity application

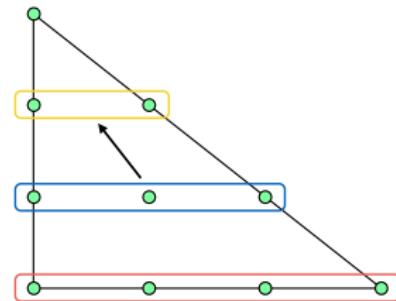
- L “lifts” numerical fluxes from faces to volume.
- Apply L_0 to face flux, extend to each “layer” of the simplex.



(a) Apply L_0 to flux to compute face output



(b) Degree reduce face nodes to compute first layer



(c) Degree reduce first layer to compute second layer

Figure: An $O(N^d)$ storage/complexity approach to applying the lift matrix.

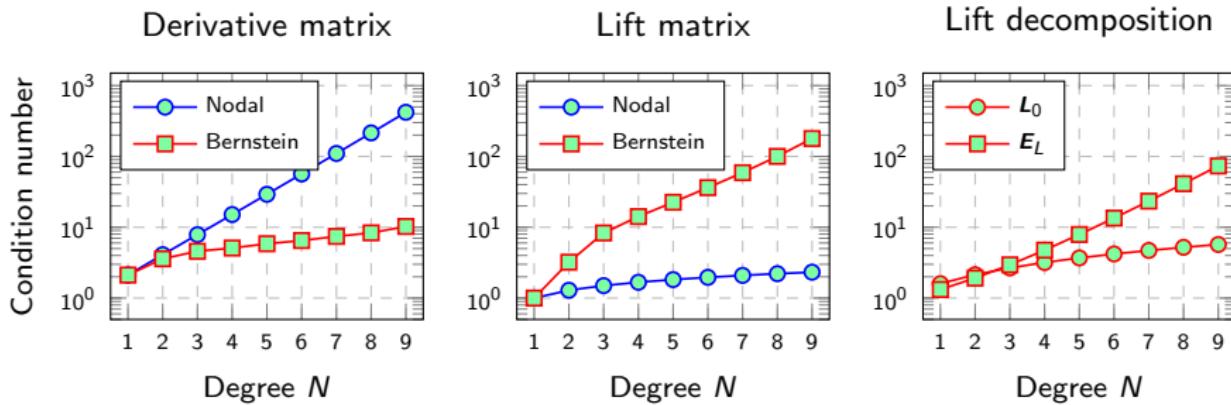
For $N < 6$, currently more efficient to treat E_L as a sparse matrix — irregular data accesses with optimal $O(N^d)$ approach.

Numerical stability of Bernstein-Bezier DG

- Conditioning of derivative, lift matrices comparable to nodal basis.

$$\kappa(\mathbf{A}) = \frac{\sigma_1}{\sigma_r}$$

- Comparable long-time growth of (single precision) numerical error.



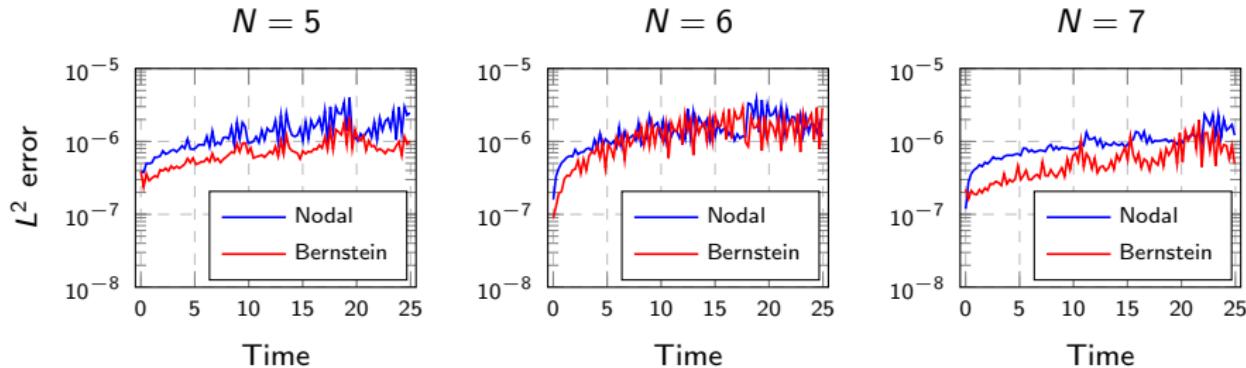
Condition numbers of matrices for nodal and Bernstein-Bezier bases.

Numerical stability of Bernstein-Bezier DG

- Conditioning of derivative, lift matrices comparable to nodal basis.

$$\kappa(\mathbf{A}) = \frac{\sigma_1}{\sigma_r}$$

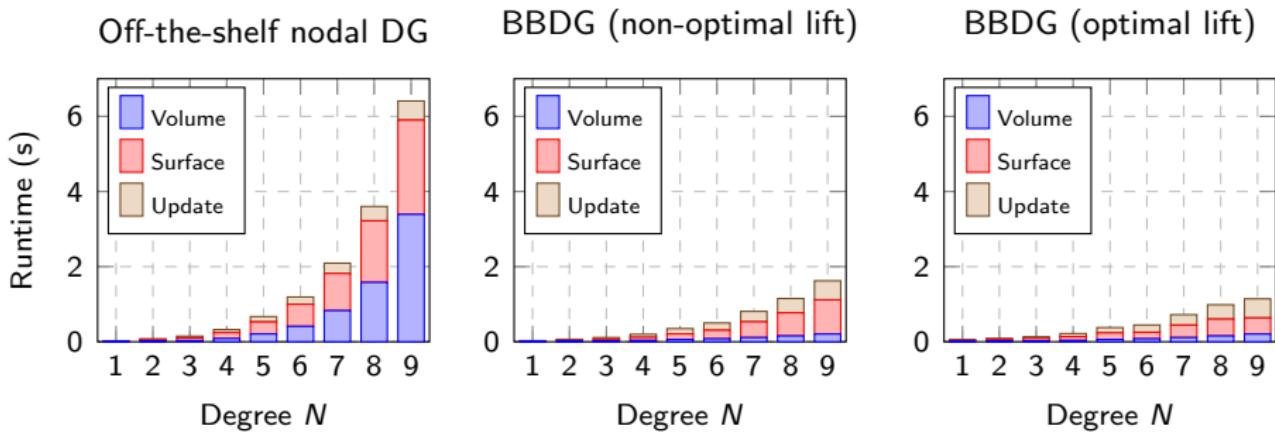
- Comparable long-time growth of (single precision) numerical error.



Evolution of L^2 error (acoustics) for nodal and Bernstein-Bezier bases.

GPU runtime comparison of BBDG and nodal DG

Bernstein-Bezier DG achieves $\approx 2\times$ speedup at moderate orders, and up to $\approx 6\times$ speedup at high orders.

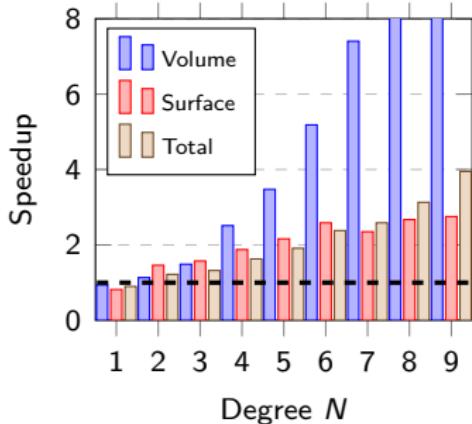


$$\underbrace{\frac{d\mathbf{u}}{dt}}_{\text{Update kernel}} = \underbrace{\mathbf{D}_x \mathbf{u}}_{\text{Volume kernel}} + \underbrace{\sum_{\text{faces}} \mathbf{L}_f}_{\text{Surface kernel}} (\text{flux}), \quad \mathbf{L}_f = \mathbf{M}^{-1} \mathbf{M}_f.$$

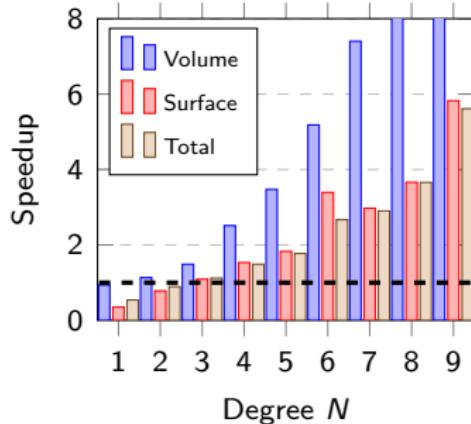
GPU runtime comparison of BBDG and nodal DG

Bernstein-Bezier DG achieves $\approx 2\times$ speedup at moderate orders, and up to $\approx 6\times$ speedup at high orders.

BB speedup (non-opt. lift) over nodal



BB speedup (optimal lift) over nodal



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

Outline

Energy stable discontinuous Galerkin formulations

- Model problem: acoustic wave equation

$$\frac{1}{c^2} \frac{\partial p}{\partial t} = \nabla \cdot \mathbf{u}, \quad \frac{\partial \mathbf{u}}{\partial t} = \nabla p$$

- (Local) formulation with penalty fluxes

$$\begin{aligned} \int_{D^k} \frac{1}{c^2} \frac{\partial p}{\partial t} q &= \int_{D^k} \nabla \cdot \mathbf{u} q + \frac{1}{2} \int_{\partial D^k} ([\![\mathbf{u}]\!] \cdot \mathbf{n} + \tau_p [\![p]\!]) q \\ \int_{D^k} \frac{\partial \mathbf{u}}{\partial t} \mathbf{v} &= \int_{D^k} \nabla p \cdot \mathbf{v} + \frac{1}{2} \int_{\partial D^k} ([\![p]\!] + \tau_u [\![\mathbf{u}]\!] \cdot \mathbf{n}) \mathbf{v} \end{aligned}$$

- High order accuracy, semi-discrete energy stability

$$\frac{\partial}{\partial t} \left(\sum_k \int_{D^k} \frac{p^2}{c^2} + |\mathbf{u}|^2 \right) = - \sum_k \int_{\partial D^k} \tau_p [\![p]\!]^2 + \tau_u [\![\mathbf{u} \cdot \mathbf{n}]\!]^2 \leq 0.$$

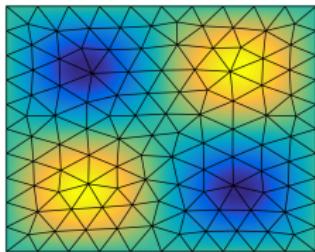
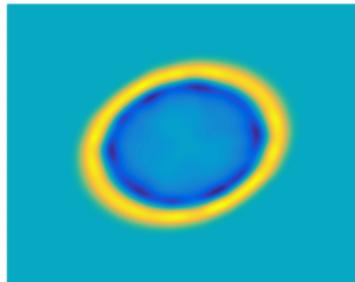
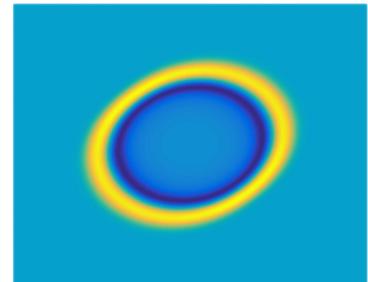
High order approximation of media and geometry

- Efficient implementation on **simplicial** meshes: c^2 piecewise constant, non-curved meshes (J, J^f piecewise constant).

$$\int_{D^k} \frac{1}{c^2} \frac{\partial p}{\partial t} q = \int_{D^k} \nabla \cdot \mathbf{u} q + \frac{1}{2} \int_{\partial D^k} (\llbracket \mathbf{u} \rrbracket \cdot \mathbf{n} + \tau_p \llbracket p \rrbracket) q$$

$$\int_{D^k} \frac{\partial \mathbf{u}}{\partial t} \mathbf{v} = \int_{D^k} \nabla p \cdot \mathbf{v} + \frac{1}{2} \int_{\partial D^k} (\llbracket p \rrbracket + \tau_u \llbracket \mathbf{u} \rrbracket \cdot \mathbf{n}) \mathbf{v}$$

- Spurious reflections for low order approximations of media, geometry.

(a) Mesh and exact c^2 (b) Piecewise const. c^2 (c) High order c^2

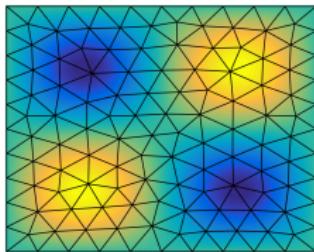
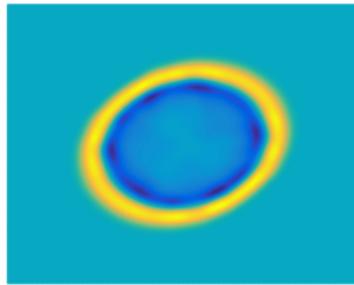
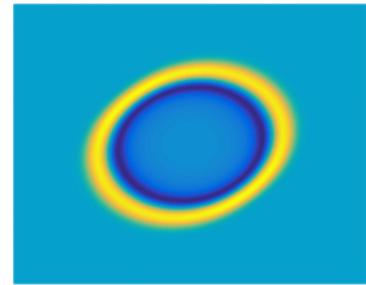
High order approximation of media and geometry

- Efficient implementation on **simplicial** meshes: c^2 piecewise constant, non-curved meshes (J, J^f piecewise constant).

$$\int_{\hat{D}} \frac{1}{c^2} \frac{\partial p}{\partial t} q J = \int_{\hat{D}} \nabla \cdot \mathbf{u} q J + \frac{1}{2} \int_{\partial \hat{D}} ([\![\mathbf{u}]\!] \cdot \mathbf{n} + \tau_p [\![p]\!]) q J^f$$

$$\int_{\hat{D}} \frac{\partial \mathbf{u}}{\partial t} \mathbf{v} J = \int_{\hat{D}} \nabla p \cdot \mathbf{v} J + \frac{1}{2} \int_{\partial \hat{D}} ([\![p]\!] + \tau_u [\![\mathbf{u}]\!] \cdot \mathbf{n}) \mathbf{v} J^f$$

- Spurious reflections for low order approximations of media, geometry.

(a) Mesh and exact c^2 (b) Piecewise const. c^2 (c) High order c^2

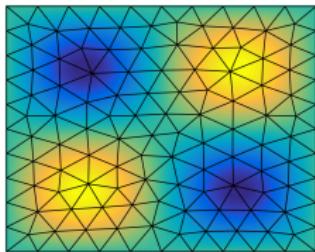
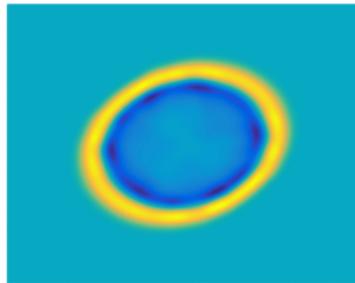
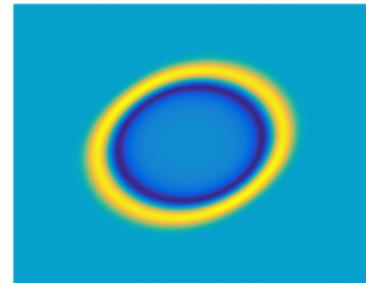
High order approximation of media and geometry

- Efficient implementation on **simplicial** meshes: c^2 piecewise constant, non-curved meshes (J, J^f piecewise constant).

$$\frac{J}{c^2} \int_{\hat{D}} \frac{\partial p}{\partial t} q = J \int_{\hat{D}} \nabla \cdot \mathbf{u} q + J^f \frac{1}{2} \int_{\partial \hat{D}} ([\![\mathbf{u}]\!] \cdot \mathbf{n} + \tau_p [\![p]\!]) q$$

$$J \int_{\hat{D}} \frac{\partial \mathbf{u}}{\partial t} \mathbf{v} = J \int_{\hat{D}} \nabla p \cdot \mathbf{v} + J^f \frac{1}{2} \int_{\partial \hat{D}} ([\![p]\!] + \tau_u [\![\mathbf{u}]\!] \cdot \mathbf{n}) \mathbf{v}$$

- Spurious reflections for low order approximations of media, geometry.

(a) Mesh and exact c^2 (b) Piecewise const. c^2 (c) High order c^2

Weighted mass matrices

- Spatially varying weights appear in DG mass matrices

$$\int_{\hat{D}} \frac{1}{c^2} \frac{\partial p}{\partial t} q J = \text{pressure RHS}, \quad \int_{\hat{D}} \frac{\partial \mathbf{u}}{\partial t} \mathbf{v} J = \text{velocity RHS}$$

- Curvilinear meshes and wave propagation in heterogeneous media

$$(\mathbf{M}_w)_{ij} = \int_{\hat{D}} \phi_i \phi_j w(x),$$

$$\frac{d}{dt} \mathbf{M}_w \mathbf{u} = \text{right hand side.}$$

- Inherits **high order accuracy** and **energy stability** with respect to a weighted L^2 norm, but requires \mathbf{M}_w^{-1} explicitly over each element.
- On-the-fly assembly + inversion or pre-computation and **storage**.

Weighted mass matrices

- Spatially varying weights appear in DG mass matrices

$$\int_{\hat{D}} \frac{1}{c^2} \frac{\partial p}{\partial t} q \mathbf{J} = \text{pressure RHS}, \quad \int_{\hat{D}} \frac{\partial \mathbf{u}}{\partial t} \mathbf{v} \mathbf{J} = \text{velocity RHS}$$

- Curvilinear** meshes and wave propagation in heterogeneous media

$$(\mathbf{M}_w)_{ij} = \int_{\hat{D}} \phi_i \phi_j \mathbf{J}(\mathbf{x}),$$

$$\frac{d}{dt} \mathbf{M}_w \mathbf{u} = \text{right hand side.}$$

- Inherits **high order accuracy** and **energy stability** with respect to a weighted L^2 norm, but requires \mathbf{M}_w^{-1} explicitly over each element.
- On-the-fly assembly + inversion or pre-computation and **storage**.

Weighted mass matrices

- Spatially varying weights appear in DG mass matrices

$$\int_{\hat{D}} \frac{1}{c^2} \frac{\partial p}{\partial t} q J = \text{pressure RHS}, \quad \int_{\hat{D}} \frac{\partial \mathbf{u}}{\partial t} \mathbf{v} J = \text{velocity RHS}$$

- Curvilinear meshes and wave propagation in **heterogeneous media**

$$(\mathbf{M}_w)_{ij} = \int_{\hat{D}} \phi_i \phi_j \frac{J}{c^2(x)},$$

$$\frac{d}{dt} \mathbf{M}_{J/c^2} \mathbf{u} = \text{right hand side.}$$

- Inherits **high order accuracy** and **energy stability** with respect to a weighted L^2 norm, but requires \mathbf{M}_w^{-1} explicitly over each element.
- On-the-fly assembly + inversion or pre-computation and **storage**.

Weighted mass matrices

- Spatially varying weights appear in DG mass matrices

$$\int_{\hat{D}} \frac{1}{c^2} \frac{\partial p}{\partial t} q J = \text{pressure RHS}, \quad \int_{\hat{D}} \frac{\partial \mathbf{u}}{\partial t} \mathbf{v} J = \text{velocity RHS}$$

- Curvilinear meshes and wave propagation in heterogeneous media

$$(\mathbf{M}_w)_{ij} = \int_{\hat{D}} \phi_i \phi_j w(x),$$

$$\frac{d}{dt} \mathbf{M}_w \mathbf{u} = \text{right hand side.}$$

- Inherits **high order accuracy** and **energy stability** with respect to a weighted L^2 norm, but requires \mathbf{M}_w^{-1} explicitly **over each element**.
- On-the-fly assembly + inversion or pre-computation and **storage**.

Weight-adjusted DG: convergence and implementation

- Weight-adjusted DG (WADG): energy stable approximation of weighted mass matrix

$$\frac{d}{dt} \mathbf{M}_w \mathbf{u} \approx \frac{d}{dt} \mathbf{M} (\mathbf{M}_{1/w})^{-1} \mathbf{M} \mathbf{u} = \text{right hand side.}$$

- WADG *a-priori* estimates: standard DG $O(h^{N+1/2})$ convergence of L^2 error based on optimal weighted projection estimate:

$$\|u - P_w u\|_{L^2} \leq Ch^{N+1} \|w\|_{W^{N+1,\infty}} \left\| \frac{\sqrt{J}}{w} \right\|_{L^\infty} \|u\|_{W^{N+1,2}}.$$

- Bypasses inverse of weighted matrix $(\mathbf{M}_w)^{-1}$

$$(\mathbf{M} (\mathbf{M}_{1/w})^{-1} \mathbf{M})^{-1} = \mathbf{M}^{-1} \mathbf{M}_{1/w} \mathbf{M}^{-1}.$$

A non-intrusive and low-storage implementation

- Operator evaluation reuses implementation for $w = 1$

$$\begin{aligned} \mathbf{M} (\mathbf{M}_{1/w})^{-1} \mathbf{M} \frac{d\mathbf{U}}{dt} &= \mathbf{A}_h \mathbf{U} \\ \rightarrow (\mathbf{M}_{1/w})^{-1} \mathbf{M} \frac{d\mathbf{U}}{dt} &= \underbrace{\mathbf{M}^{-1} \mathbf{A}_h \mathbf{U}}_{\text{RHS for } w=1} \end{aligned}$$

- Low storage: matrix-free application of $\mathbf{M}^{-1} \mathbf{M}_{1/w}$.

$$(\mathbf{M})^{-1} \mathbf{M}_{1/w} \text{RHS} = \underbrace{\widehat{\mathbf{M}}^{-1} \mathbf{V}_q^T W \text{diag}(1/w) \mathbf{V}_q}_{P_q} (\text{RHS}).$$

- Non-intrusive: modify RHS locally before update. For non-curved meshes, can combine with *optimal complexity* RHS evaluation.¹

Weight-adjusted DG: not locally conservative

- **Con:** loss of local conservation for $w(x) \notin P^N$!

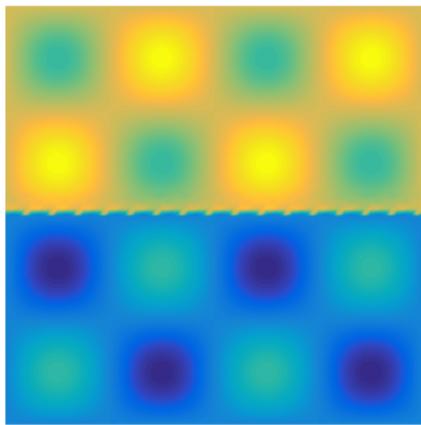
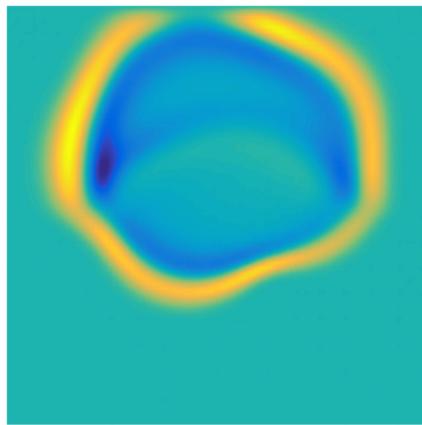
- **Pro:** superconvergence of conservation error

$$\text{Conservation error} \leq Ch^{2N+2} \|w\|_{W^{N+1,\infty}} \|p\|_{W^{N+1,2}}$$

where C depends on mesh quality and max/min values of w .

- **Pro:** can restore local conservation with rank-1 update (Shermann-Morrison).

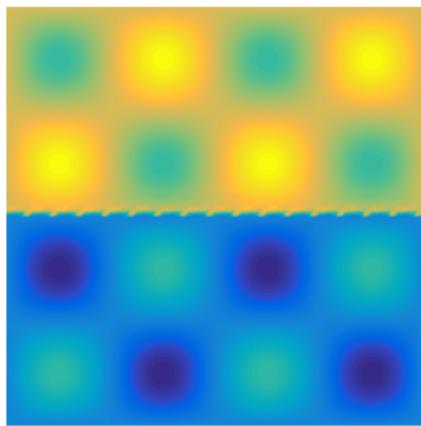
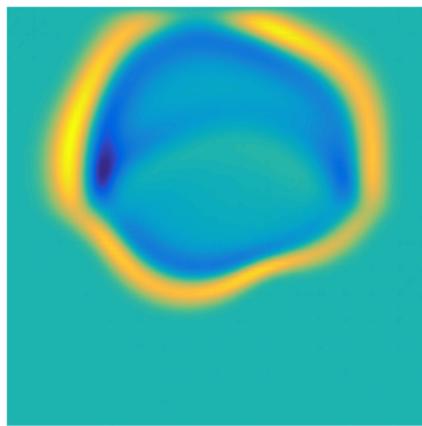
Acoustic wave equation: heterogeneous media

(a) $c^2(x, y)$ 

(b) Standard DG

Figure: Standard vs. weight-adjusted DG with spatially varying c^2 containing both smooth variations and a discontinuity.

Acoustic wave equation: heterogeneous media

(a) $c^2(x, y)$ 

(b) Weighted-adjusted DG

Figure: Standard vs. weight-adjusted DG with spatially varying c^2 containing both smooth variations and a discontinuity.

Acoustics, variable coefficients: L^2 errors

Smooth wavefield $c^2(x, y) = 1 + \frac{1}{2} \sin(\pi x) \sin(\pi y)$

	$h = 1$	$h = 1/2$	$h = 1/4$	$h = 1/8$
DG $N = 1$	2.13e-01	6.25e-02	1.64e-02	4.19e-03
WADG $N = 1$	2.05e-01	5.99e-02	1.62e-02	4.18e-03
DG $N = 2$	3.01e-02	3.60e-03	4.21e-04	5.07e-05
WADG $N = 2$	2.89e-02	3.54e-03	4.18e-04	5.07e-05
DG $N = 3$	6.10e-03	3.33e-04	2.04e-05	1.22e-06
WADG $N = 3$	8.69e-03	3.47e-04	2.03e-05	1.22e-06
DG $N = 4$	6.61e-04	2.12e-05	6.39e-07	1.94e-08
WADG $N = 4$	1.09e-03	2.27e-05	6.30e-07	1.93e-08

Table: Convergence of standard, weight-adjusted DG to a manufactured solution.

Acoustics, variable coefficients: L^2 errors

Smooth wavefield $c^2(x, y) = 1 + \frac{1}{2} \sin(\pi x) \sin(\pi y)$

	$h = 1$	$h = 1/2$	$h = 1/4$	$h = 1/8$
DG $N = 1$	2.48e-01	7.58e-02	1.69e-02	4.46e-03
WADG $N = 1$	2.50e-01	7.72e-02	1.69e-02	4.47e-03
DG $N = 2$	5.95e-02	9.95e-03	1.10e-03	1.22e-04
WADG $N = 2$	6.09e-02	1.02e-02	1.10e-03	1.22e-04
DG $N = 3$	2.29e-02	1.98e-03	9.52e-05	6.56e-06
WADG $N = 3$	1.98e-02	1.98e-03	9.52e-05	6.56e-06
DG $N = 4$	4.90e-03	3.01e-04	1.78e-05	7.27e-07
WADG $N = 4$	4.64e-03	3.02e-04	1.78e-05	7.28e-07

Table: Convergence to a reference solution ($N = 100$ spectral method).

Acoustics, variable coefficients: convergence

	$N = 1$	$N = 2$	$N = 3$	$N = 4$
DG	1.9220	3.0752	4.0440	5.0446
WADG	1.9211	3.0629	4.0752	5.0990

(a) Rates of convergence to manufactured solution

	$N = 1$	$N = 2$	$N = 3$	$N = 4$
DG	1.8256	3.1796	3.8589	4.6171
WADG	1.8425	3.1807	3.8583	4.6128

(b) Rates of convergence to reference solution

Observed L^2 rates between optimal $O(h^{N+1})$ and estimated $O(h^{N+1/2})$.

Outline

Weight-adjusted DG for curvilinear meshes

- Weight-adjusted L^2 projection \tilde{P}_N on curved domains

$$\tilde{P}_N(u) := P_N \left(\frac{1}{J} P_N(uJ) \right).$$

where P_N is the L^2 projection onto the reference element.

- L^2 estimates for weight-adjusted projection:

$$\|u - \tilde{P}_N u\|_{L^2(D^k)} \lesssim \left\| \frac{1}{\sqrt{J}} \right\|_{L^\infty}^2 \|J\|_{W^{N+1,\infty}(D^k)} h^{N+1} \|u\|_{W^{N+1,2}(D^k)}.$$

- High order Sobolev norm of J - implies that convergence can suffer for non-smooth mappings.

Behavior of weight-adjusted L^2 projection

Comparison with L^2 projection and Low-Storage Curvilinear DG

$$\tilde{\phi}_i = \frac{\phi_i}{\sqrt{J}}, \quad \mathbf{M}_{ij} = \int_{D^k} \tilde{\phi}_j \tilde{\phi}_i J = \int_{\hat{D}} \phi_j \phi_i = \widehat{\mathbf{M}}_{ij}.$$

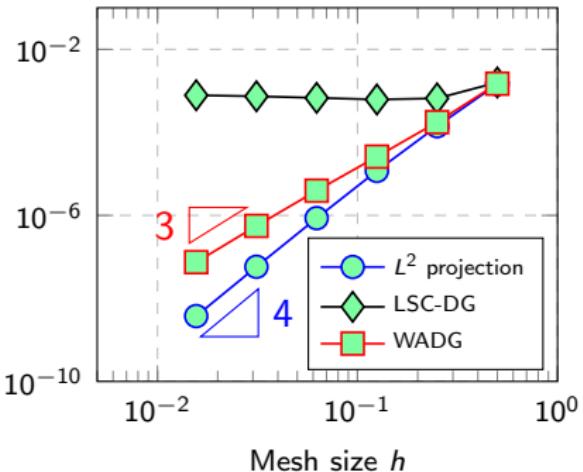
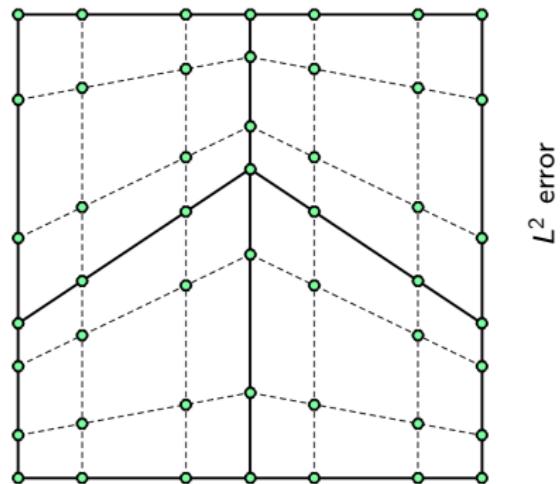


Figure: Arnold-type mesh with $\|J\|_{W^{N+1,\infty}} = O(h^{-1})$ for $N = 3$.

Behavior of weight-adjusted L^2 projection

Comparison with L^2 projection and Low-Storage Curvilinear DG

$$\tilde{\phi}_i = \frac{\phi_i}{\sqrt{J}}, \quad \mathbf{M}_{ij} = \int_{D^k} \tilde{\phi}_j \tilde{\phi}_i J = \int_{\hat{D}} \phi_j \phi_i = \widehat{\mathbf{M}}_{ij}.$$

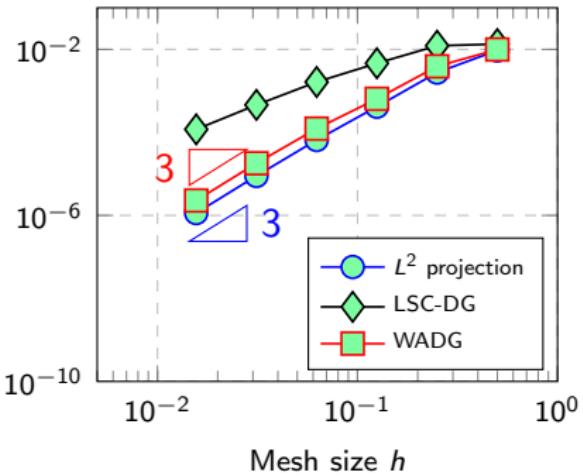
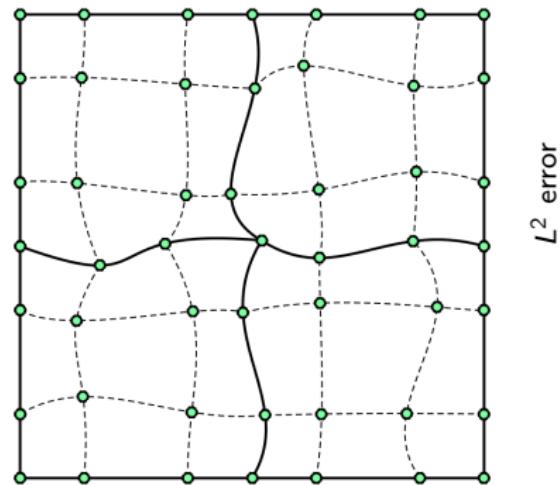


Figure: Curvilinear mesh constructed through random perturbation for $N = 3$.

Behavior of weight-adjusted L^2 projection

High order convergence **slowed** by growth of $\|J\|_{W^{N+1,\infty}} = O(h^N)$.

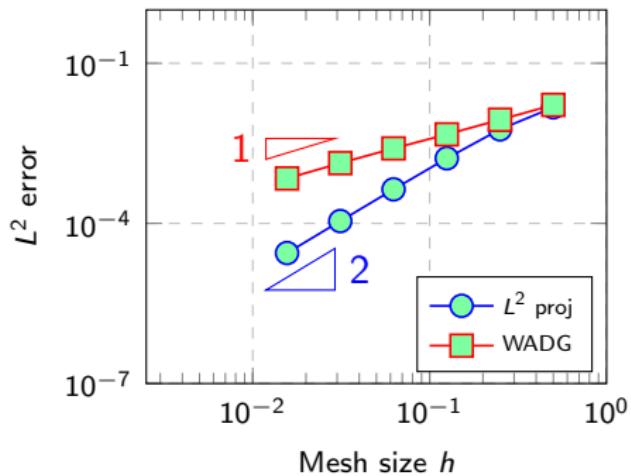
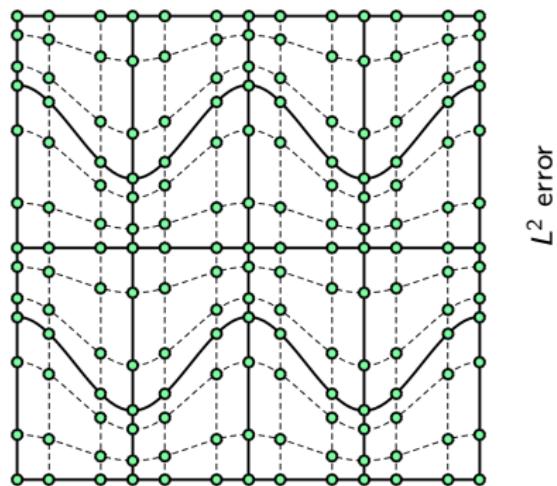


Figure: Moderately warped curved Arnold-type mesh for $N = 3$.

Behavior of weight-adjusted L^2 projection

High order convergence is **stalled** by growth of $\|J\|_{W^{N+1,\infty}} = O(h^{N+1})$.

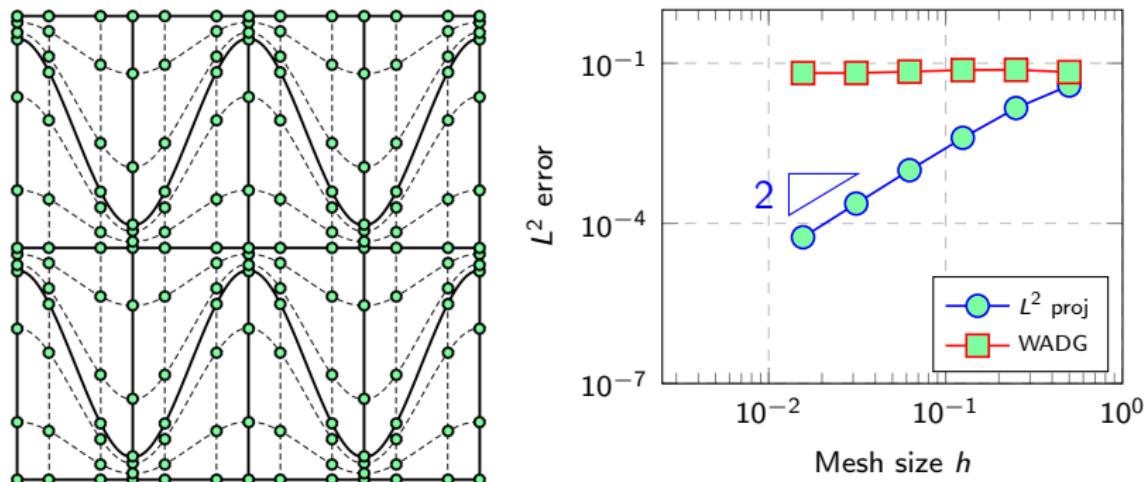
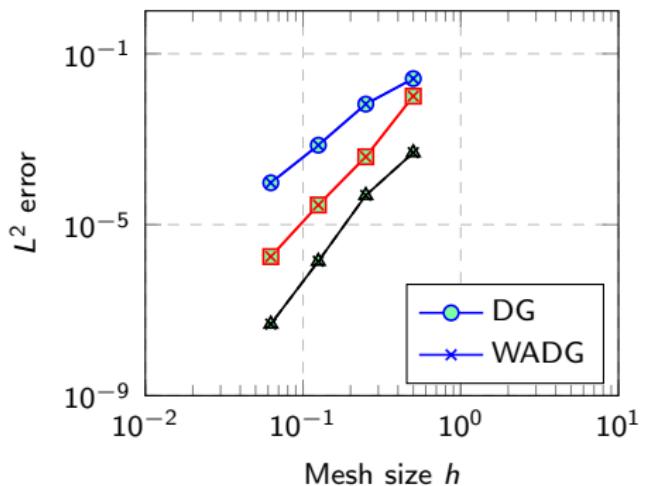
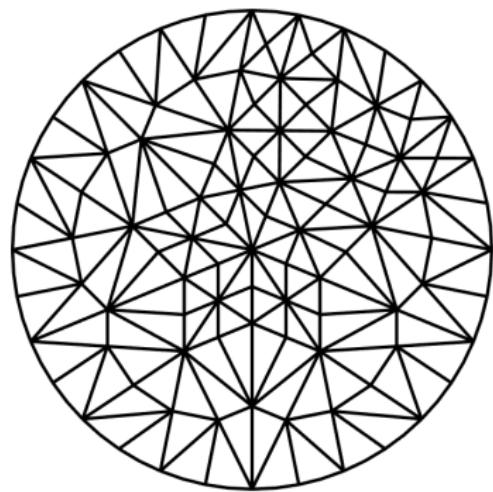


Figure: Heavily warped curved Arnold-type mesh for $N = 3$.

Curvilinear meshes: two-dimensional verification

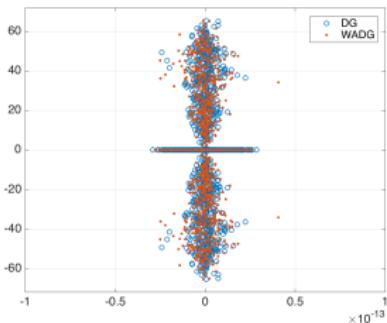
Energy stability: quadrature-based skew-symmetric formulation.



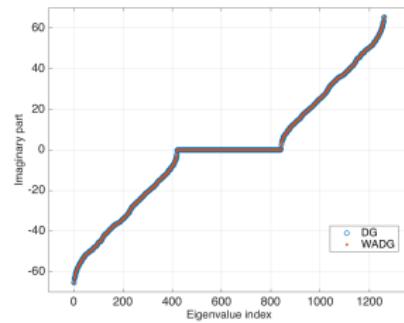
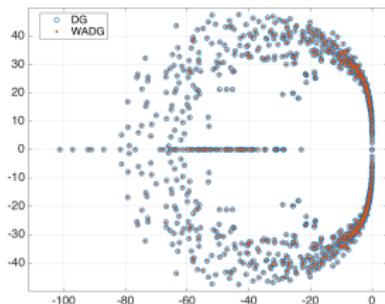
(a) L^2 errors for $N = 2, 3, 4$

Figure: Optimal L^2 convergence rates observed for curvilinear meshes.

Curvilinear meshes: DG eigenvalues (circular domain)

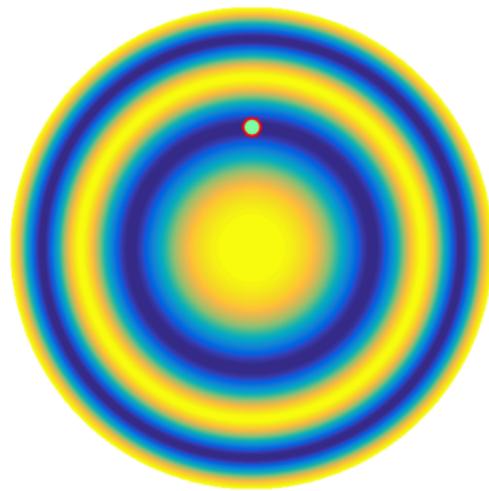


(a) Central fluxes

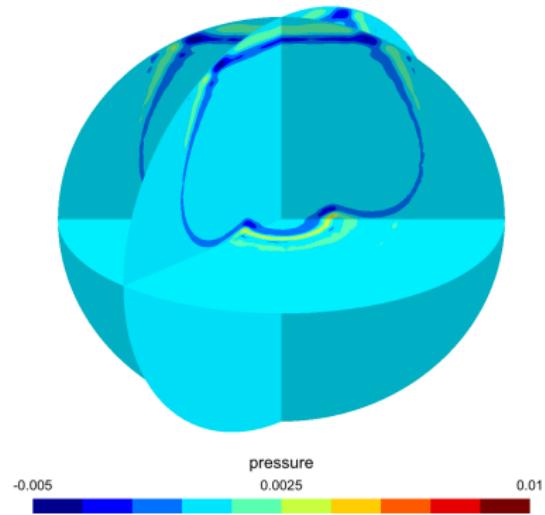
(b) $\text{Im}(\lambda_i)$ for central fluxes

(c) Upwind fluxes

Curved meshes + heterogeneous media



(a) Wavespeed $c^2(x)$



(b) Pressure isovalues at $t = .6$

Can incorporate heterogeneous media with curved elements at no additional cost.

Time-domain spline methods

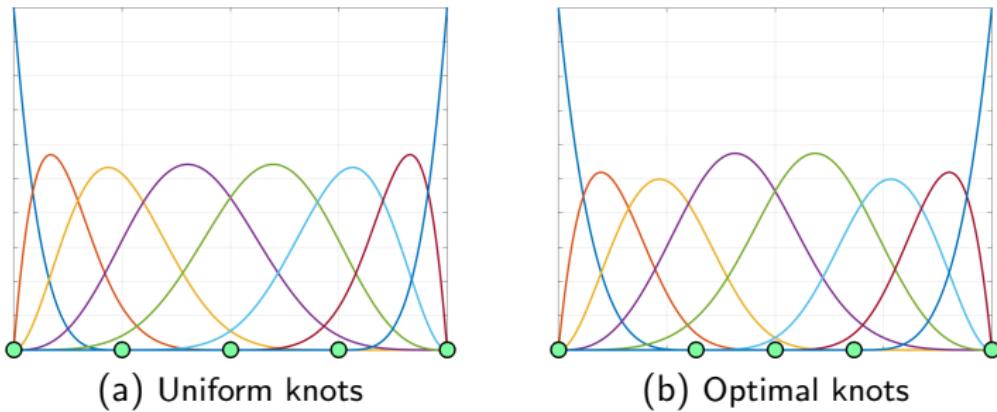


Figure: B-spline bases with uniform and optimal knot vectors ($N = 4, K = 4$).

- Optimal knots (at roots of specific eigenfunctions) minimize n -width.
 - We approximate optimal knots using heuristic “smoothing”.
 - Lack of mass lumping: expensive to apply M^{-1} .

Melkman and Micchelli 1978. Spline spaces are optimal for L^2 n -width.

Restoring Kronecker structure to M^{-1}

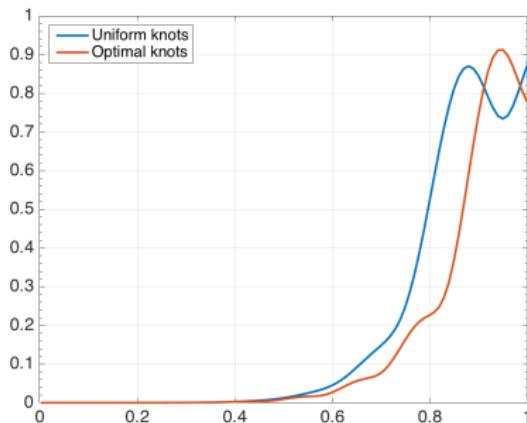
- Loss of Kronecker product structure in spline mass matrix inverses.
- WADG recovers tensor product: application of $M_{1/J}$, \widehat{M}^{-1}

$$\begin{aligned} M_J^{-1} &\approx \widehat{M}^{-1} M_{1/J} \widehat{M}^{-1} \\ \widehat{M}^{-1} &= \widehat{M}_{1D}^{-1} \otimes \widehat{M}_{1D}^{-1} \otimes \widehat{M}_{1D}^{-1}. \end{aligned}$$

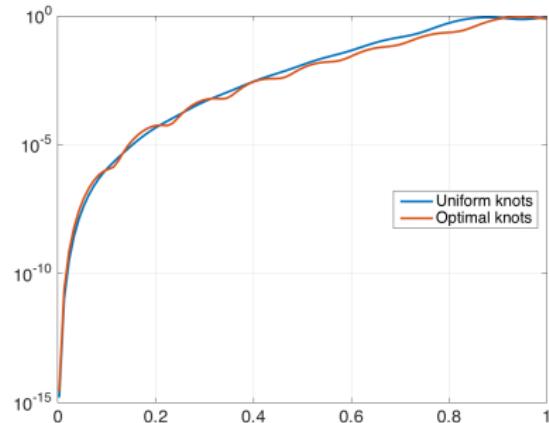
- Maintains **provable** energy stability for general geometric mappings.
- Same approach has been used for Galerkin difference methods.

Approximation properties of optimal splines

- Improved (pre-asymptotic?) resolution for both oscillatory functions and non-affine mappings.
- Multi-patch IGA: optimal h -convergence rates (acoustics).



(a) L^2 error vs wavelengths (k) per dof



(b) Semi-log scale

Figure: L^2 error in approximating $u(x) = \cos\left(\frac{k\pi x}{2}\right)$ with $N = 4, K = 16$.

Approximation properties of optimal splines

- Improved (pre-asymptotic?) resolution for both oscillatory functions and non-affine mappings.
- Multi-patch IGA: optimal h -convergence rates (acoustics).

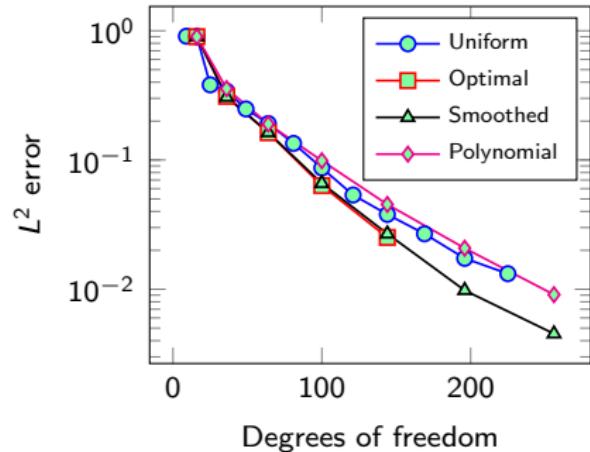
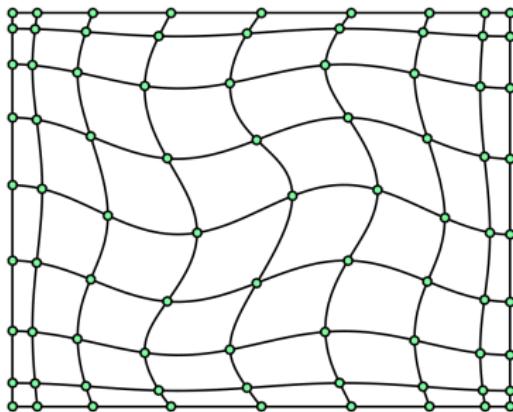
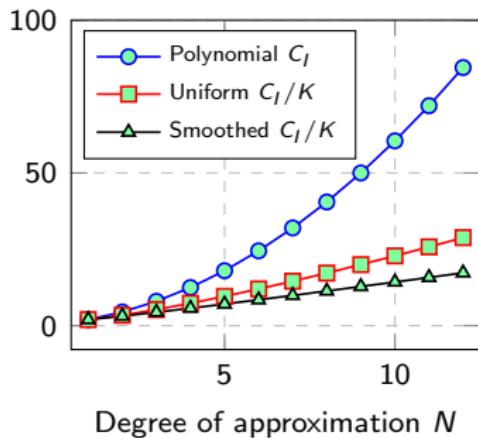
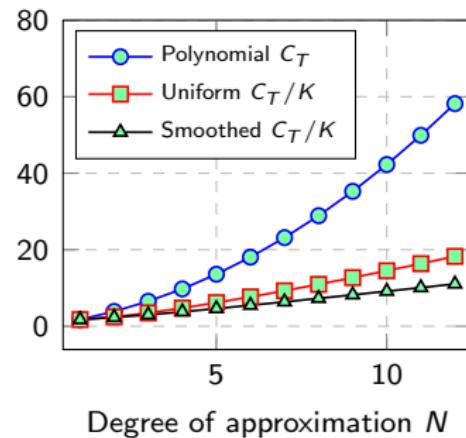


Figure: L^2 errors for $\cos\left(\frac{3\pi x}{2}\right)\cos\left(\frac{3\pi y}{2}\right)$ under degree refinement $N = 2, \dots, 8$, using spline spaces with $K = N$ elements.

Application to IGA and time-domain wave propagation

- Bound $\rho(\mathbf{A}_h)$ using generalized Rayleigh quotients (Bendixon-Hirsch): depends on h and **constants** C_T, C_I .
- CFL: $O(h/N^2)$ for polynomials, $O(h/N)$ for splines if $h \geq O(1/N)$.

(a) Inverse inequality, $K = 2N$ (b) Trace inequality, $K = 2N$

Outline

Matrix-valued weights and elastic wave propagation

- Symmetric velocity-stress form of linear elasticity (\mathbf{A}_i constant)

$$\rho \frac{\partial \mathbf{v}}{\partial t} = \sum_{i=1}^d \mathbf{A}_i^T \frac{\partial \boldsymbol{\sigma}}{\partial \mathbf{x}_i}, \quad \mathbf{C}^{-1} \frac{\partial \boldsymbol{\sigma}}{\partial t} = \sum_{i=1}^d \mathbf{A}_i \frac{\partial \mathbf{v}}{\partial \mathbf{x}_i}.$$

- DG formulation based on penalty fluxes, matrix-weighted mass matrix

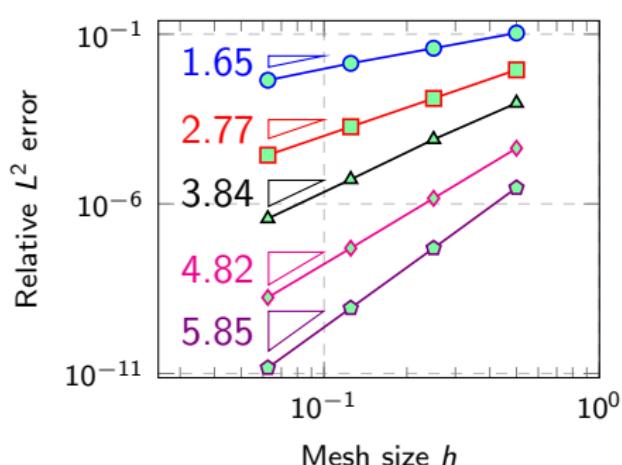
$$(\mathbf{M}_{\mathbf{C}^{-1}})^{-1} = \begin{pmatrix} \mathbf{M}_{C_{11}^{-1}} & \dots & \mathbf{M}_{C_{1d}^{-1}} \\ \vdots & \ddots & \vdots \\ \mathbf{M}_{C_{d1}^{-1}} & \dots & \mathbf{M}_{C_{dd}^{-1}} \end{pmatrix}$$

- Weight-adjusted approximation for \mathbf{C}^{-1} decouples into components

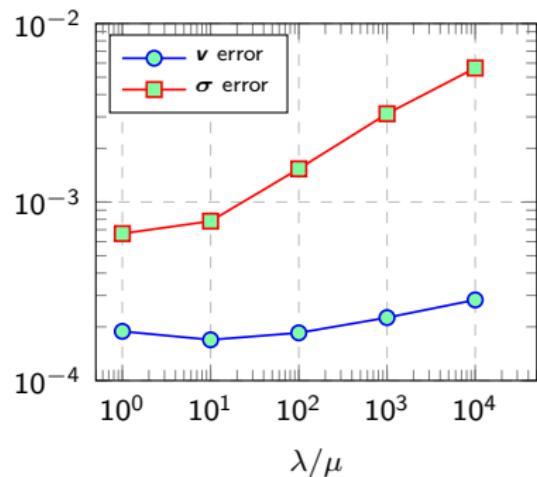
$$\mathbf{M}_{\mathbf{C}^{-1}}^{-1} \approx (\mathbf{I} \otimes \mathbf{M}^{-1}) \mathbf{M}_{\mathbf{C}} (\mathbf{I} \otimes \mathbf{M}^{-1}).$$

Elastic wave propagation: convergence

- Convergence for harmonic oscillation, Rayleigh, Lamb, and Stoneley waves: between $O(h^{N+1})$ and $O(h^{N+1/2})$.
- σ error grows as $\|\mathbf{C}^{-1}\| \rightarrow \infty$ (e.g. incompressible limit $\lambda/\mu \rightarrow \infty$).



(a) Stoneley wave

(b) $\|\mathbf{C}^{-1}\| \rightarrow \infty$, $N = 3, h = 1/8$.

Elastic wave propagation: anisotropy

No change in implementation for anisotropy - fluxes independent of C .

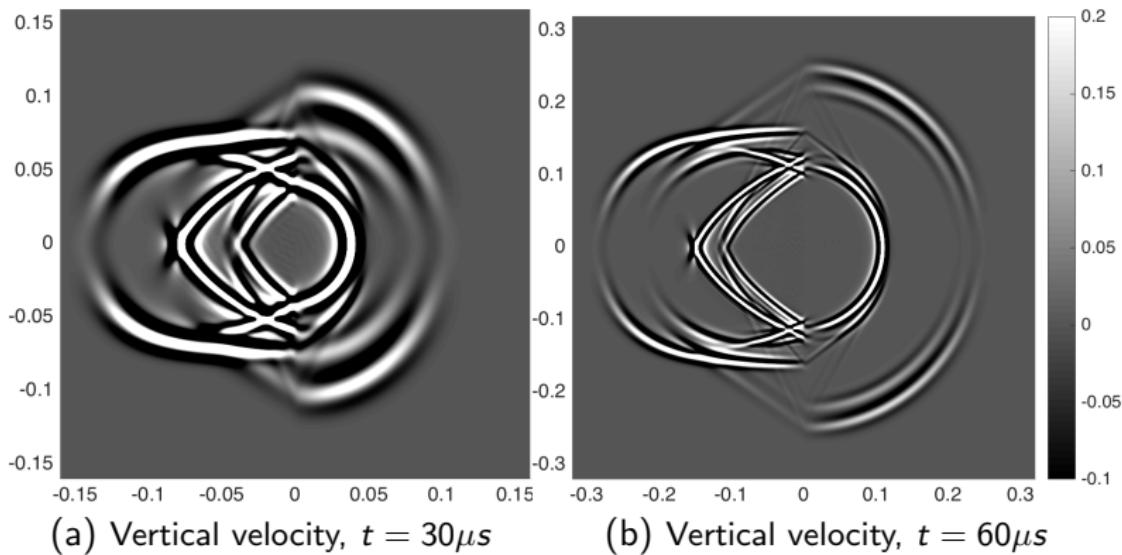


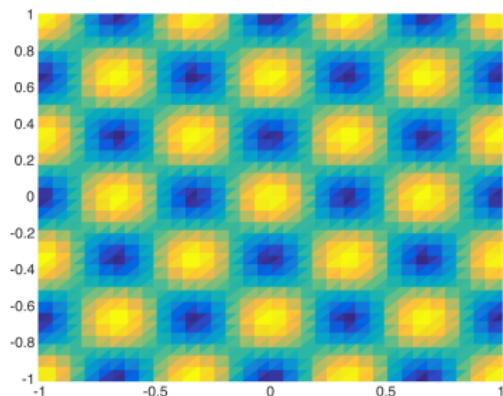
Figure: Heterogeneous media: transverse isotropy ($x < 0$) and isotropy ($x > 0$).

Elastic wave propagation: acoustic-elastic coupling

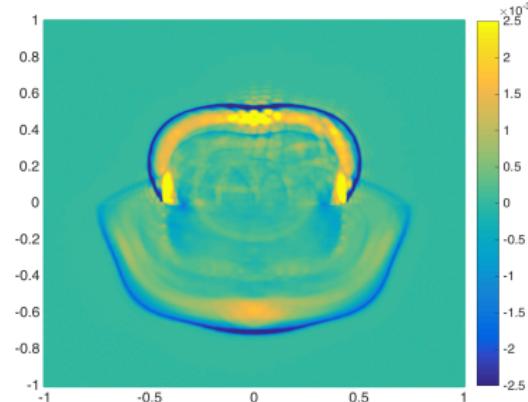
- Interface jumps become residuals of continuity conditions:

$$\boldsymbol{\sigma} \cdot \mathbf{n} = p\mathbf{n}, \quad \mathbf{v} \cdot \mathbf{n} = \mathbf{u} \cdot \mathbf{n}.$$

- Energy stable for arbitrary heterogeneous media.



(a) Low order $c^2(x), \mu(x)$



(b) $\text{tr}(\boldsymbol{\sigma})$

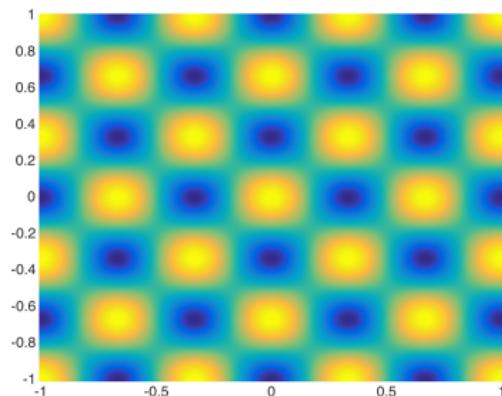
Figure: Acoustic-elastic waves from a Ricker pulse ($N = 10, h = 1/16$).

Elastic wave propagation: acoustic-elastic coupling

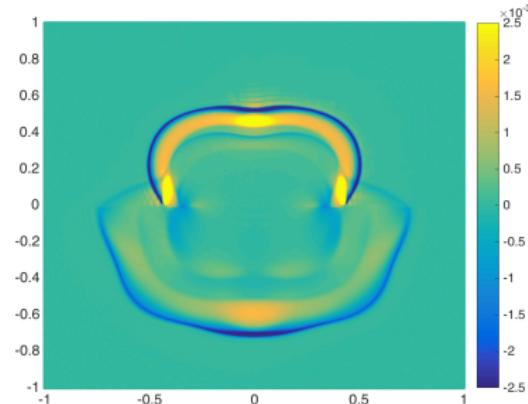
- Interface jumps become residuals of continuity conditions:

$$\boldsymbol{\sigma} \cdot \mathbf{n} = p\mathbf{n}, \quad \mathbf{v} \cdot \mathbf{n} = \mathbf{u} \cdot \mathbf{n}.$$

- Energy stable for arbitrary heterogeneous media.



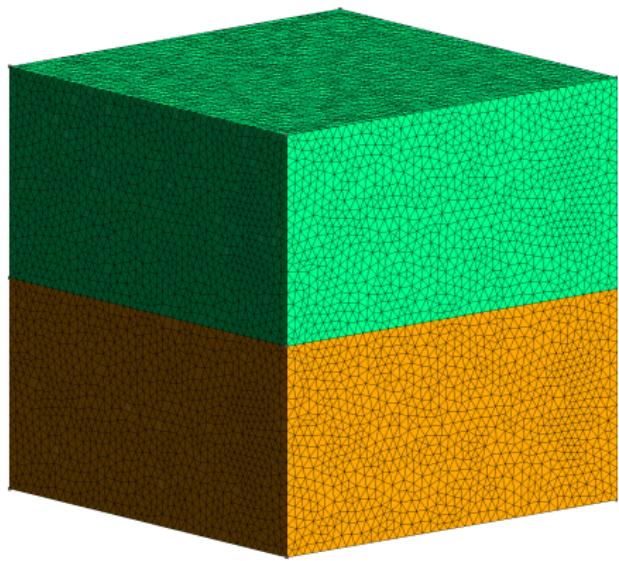
(a) High order $c^2(x), \mu(x)$



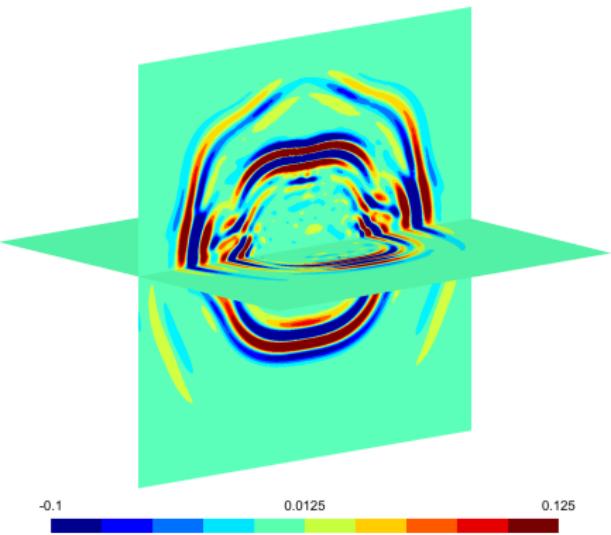
(b) $\text{tr}(\boldsymbol{\sigma})$

Figure: Acoustic-elastic waves from a Ricker pulse ($N = 10, h = 1/16$).

Elastic wave propagation: 3D isotropic media



(a) Computational mesh



(b) Piecewise constant $C(x)$

Figure: $\text{tr}(\sigma)$ with $\mu(x) = 1 + H(y) + \frac{1}{2} \cos(3\pi x) \cos(3\pi y) \cos(3\pi z)$, $N = 5$.

Elastic wave propagation: 3D isotropic media

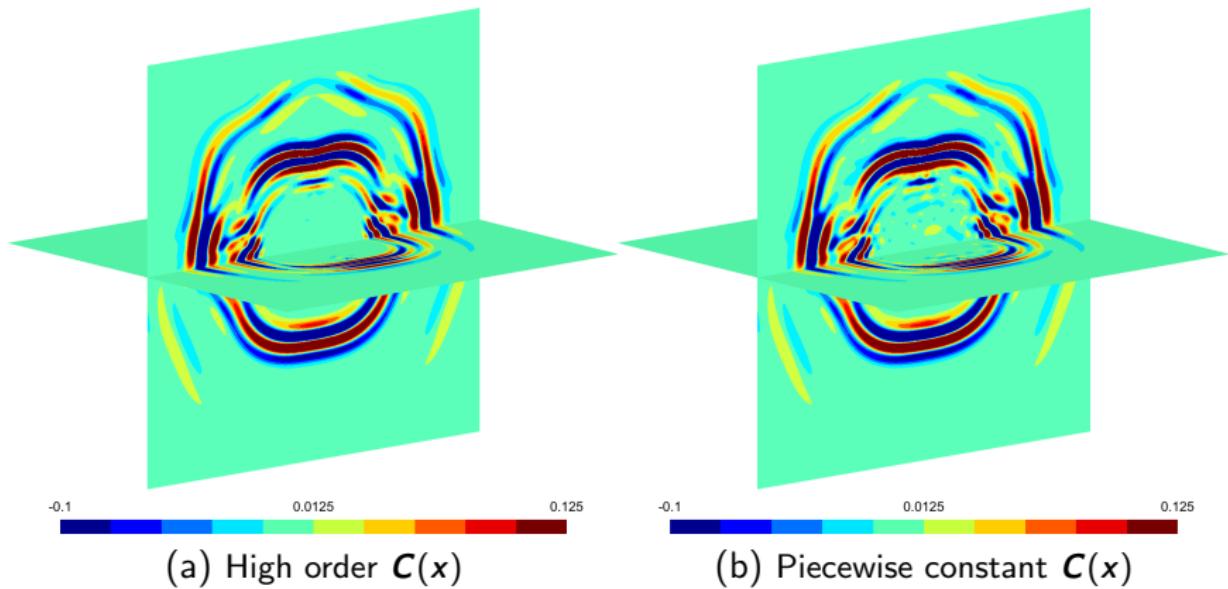


Figure: $\text{tr}(\sigma)$ with $\mu(x) = 1 + H(y) + \frac{1}{2} \cos(3\pi x) \cos(3\pi y) \cos(3\pi z)$, $N = 5$.

Summary and acknowledgements

- **Sparsity** of Bernstein-Bezier DG: efficiency at high orders on GPUs.
- Weight-adjusted DG (WADG): low-storage methods for heterogeneous media and curvilinear meshes.
- Future work:
 - Exploit structure of WADG under Bernstein-Bezier basis.
 - WADG with singular weights.

Thanks to TOTAL E&P Research and Technology USA
for their support of this work.

Chan 2017. Weight-adjusted DG methods: matrix-valued weights and elastic wave prop. in heterogeneous media (arXiv).

Chan, et al. 2016. Weight-adjusted DG methods: wave propagation in heterogeneous media (arXiv).

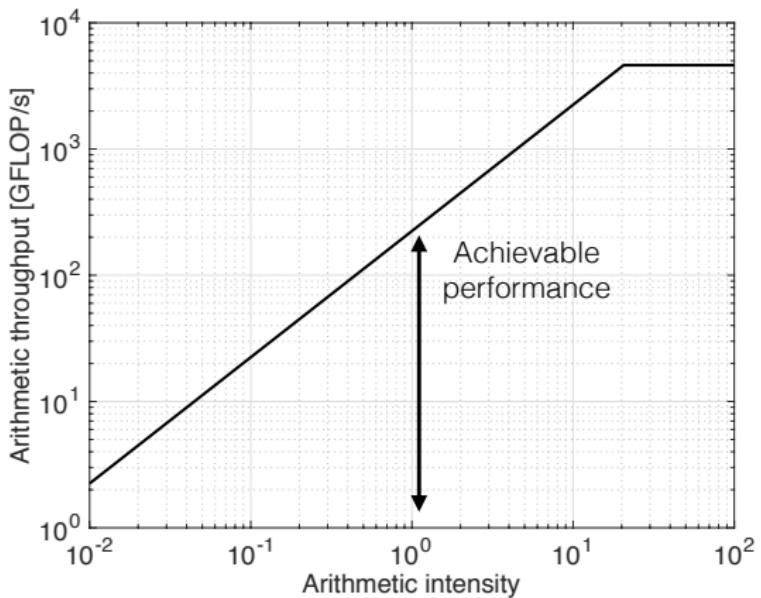
Chan, et al. 2016. Weight-adjusted DG methods: curvilinear meshes (arXiv).

Chan, Warburton 2015. GPU-accelerated Bernstein-Bezier DG methods for wave problems (SISC).

Additional slides

Roofline model: estimating computational efficiency

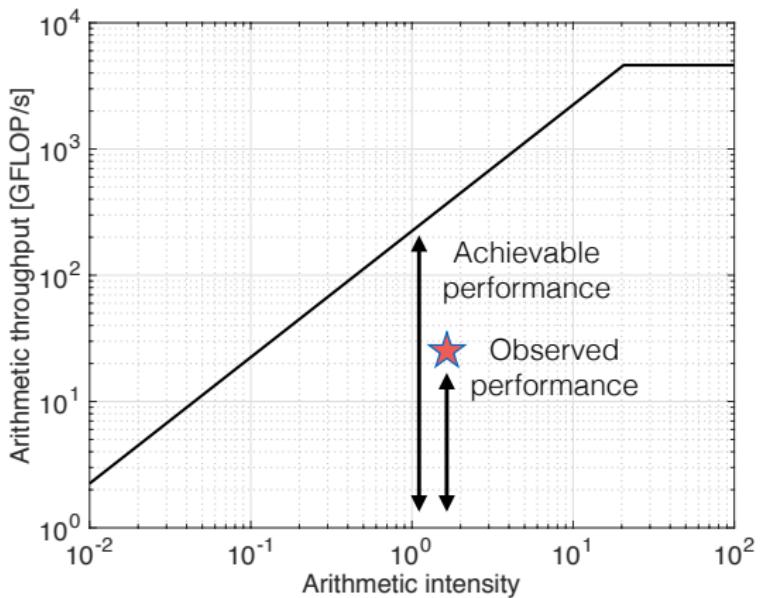
- Arithmetic intensity: floating-point operations per byte of data.
- Computational efficiency: ratio of observed/achievable performance.



Williams, Waterman, Patterson 2009. Roofline: an insightful visual performance model for multicore architectures.

Roofline model: estimating computational efficiency

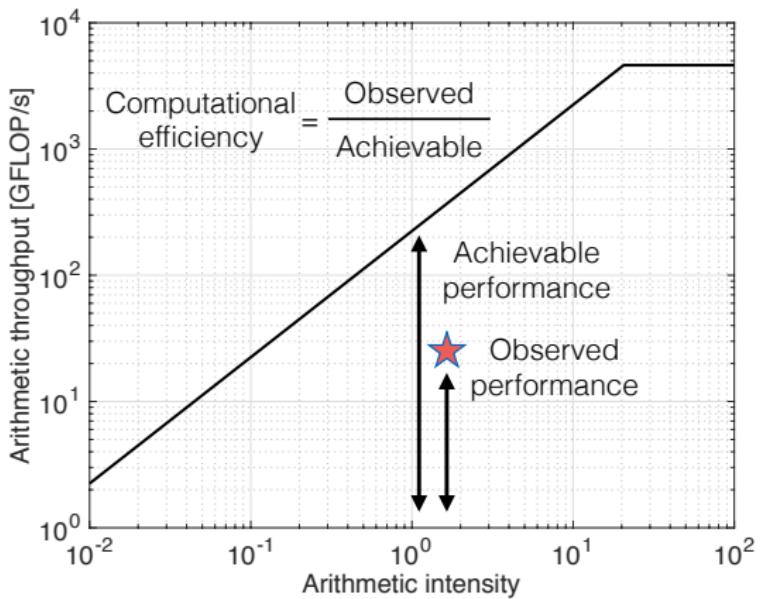
- Arithmetic intensity: floating-point operations per byte of data.
- Computational efficiency: ratio of observed/achievable performance.



Williams, Waterman, Patterson 2009. Roofline: an insightful visual performance model for multicore architectures.

Roofline model: estimating computational efficiency

- Arithmetic intensity: floating-point operations per byte of data.
- Computational efficiency: ratio of observed/achievable performance.

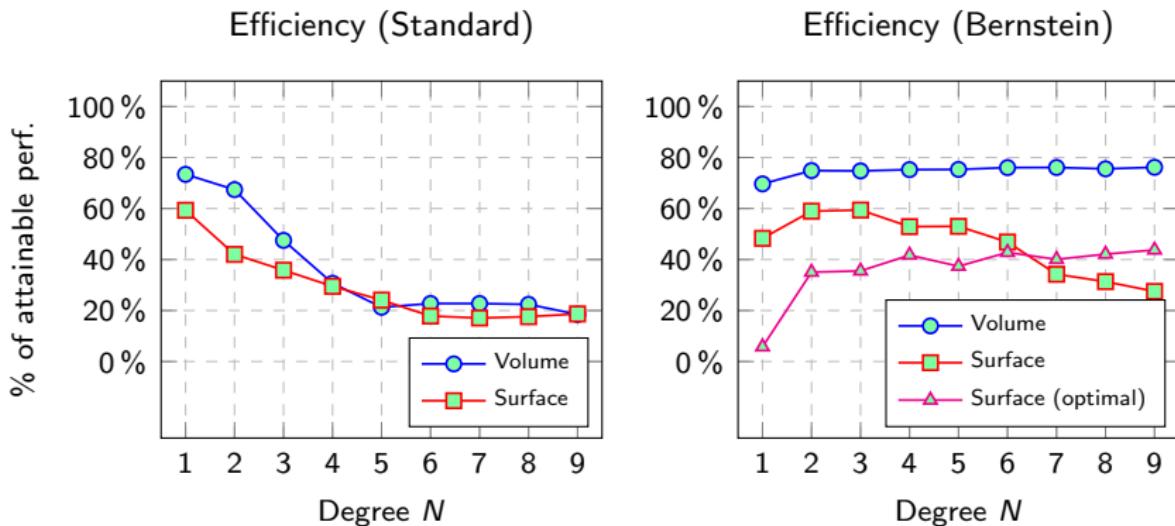


Williams, Waterman, Patterson 2009. Roofline: an insightful visual performance model for multicore architectures.

Performance comparisons of Bernstein-Bezier DG

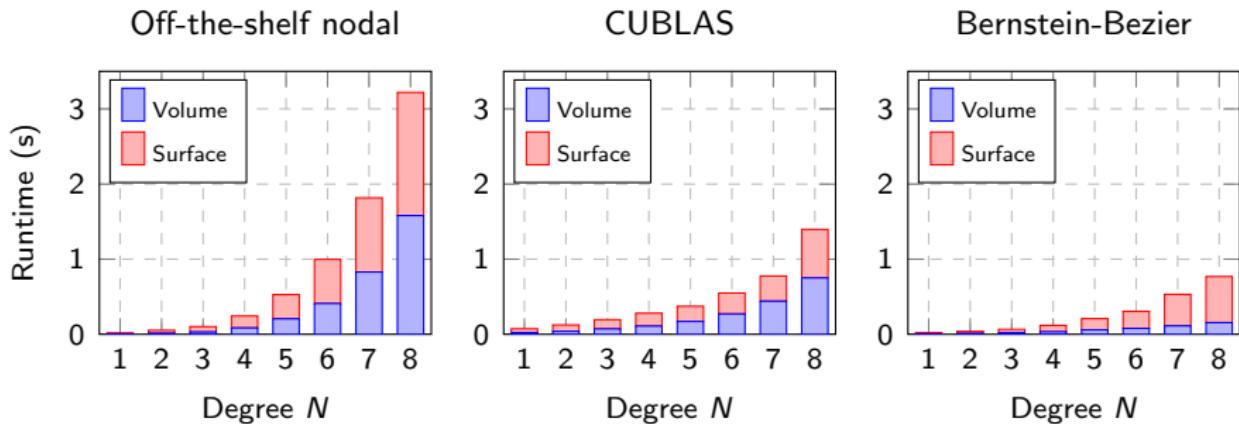
Bernstein-Bezier DG: standard implementation, sparse matrices.

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



Bernstein-Bezier compared to CUBLAS

Bernstein-Bezier DG achieves $\approx 4 \times$ speedup at low-moderate orders, and $1.5 - 2 \times$ speedup at high orders.

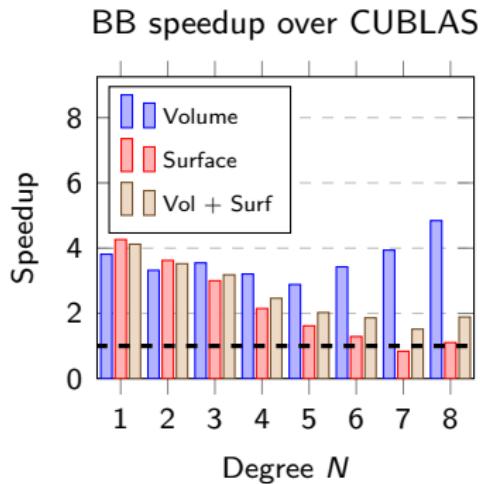


$$\frac{d\mathbf{u}}{dt} = \underbrace{\mathbf{D}_x \mathbf{u}}_{\text{Volume kernel}} + \underbrace{\sum_{\text{faces}} \mathbf{L}_f}_{\text{Surface kernel}} (\text{flux}), \quad \mathbf{L}_f = \mathbf{M}^{-1} \mathbf{M}_f.$$

Update kernel

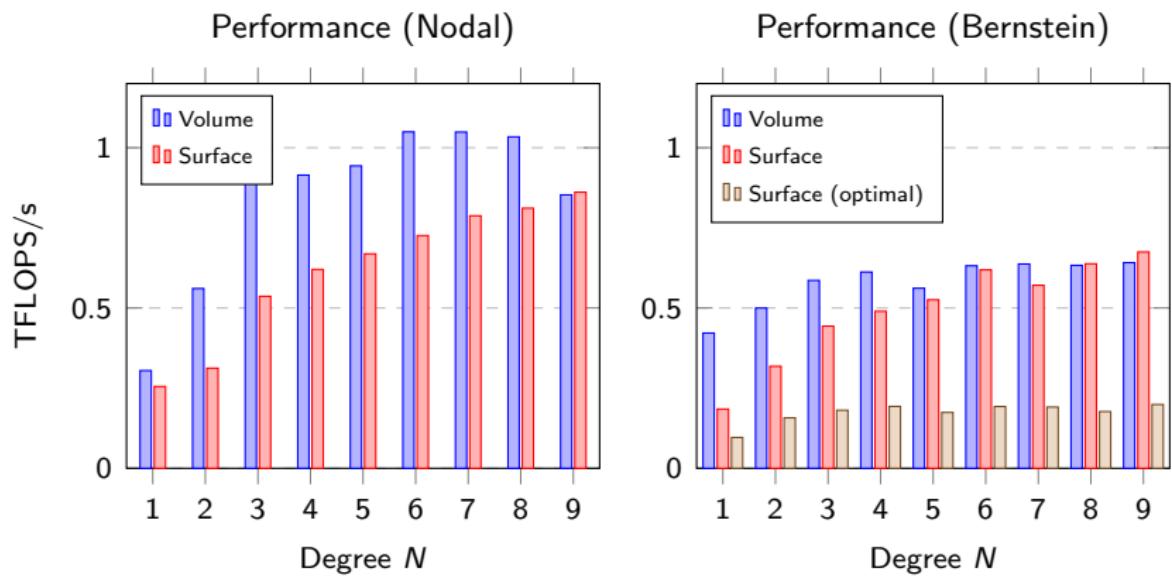
Bernstein-Bezier compared to CUBLAS

Bernstein-Bezier DG achieves $\approx 4 \times$ speedup at low-moderate orders, and $1.5 - 2 \times$ speedup at high orders.

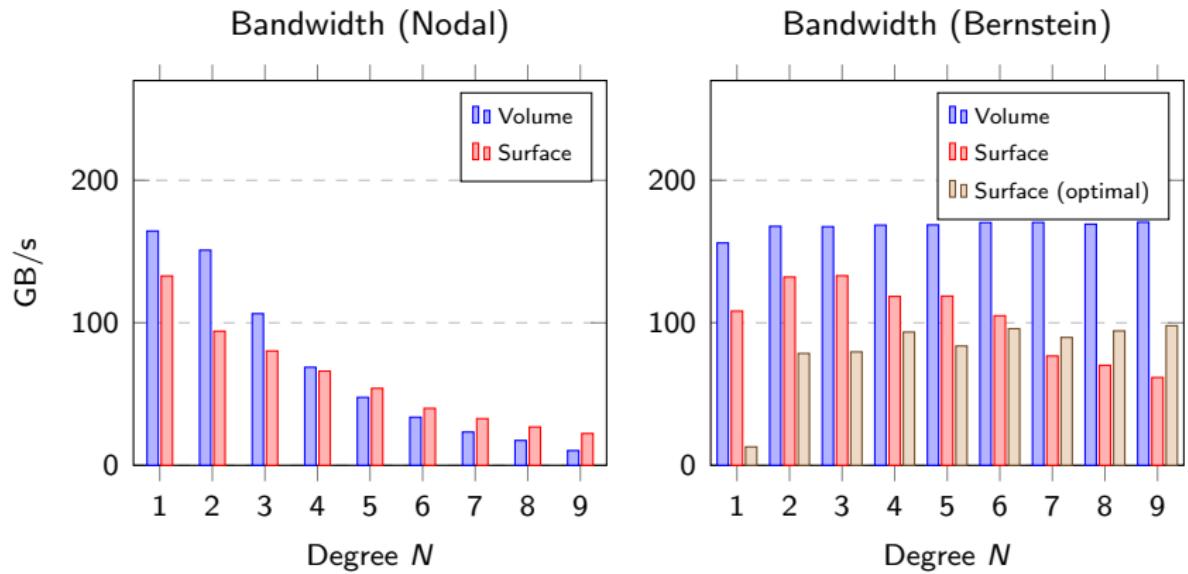


$$\underbrace{\frac{d\mathbf{u}}{dt}}_{\text{Update kernel}} = \underbrace{\mathbf{D}_x \mathbf{u}}_{\text{Volume kernel}} + \underbrace{\sum_{\text{faces}} \mathbf{L}_f}_{\text{Surface kernel}} (\text{flux}), \quad \mathbf{L}_f = \mathbf{M}^{-1} \mathbf{M}_f.$$

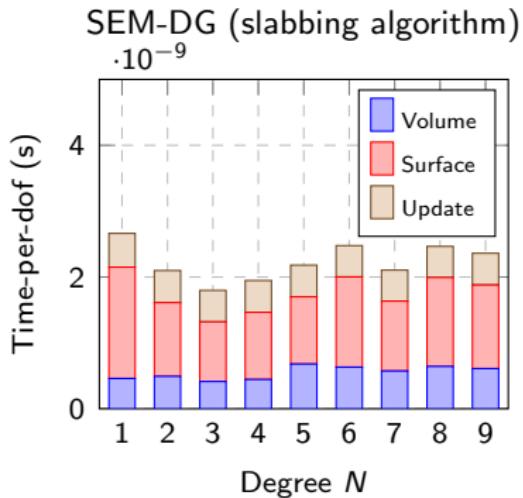
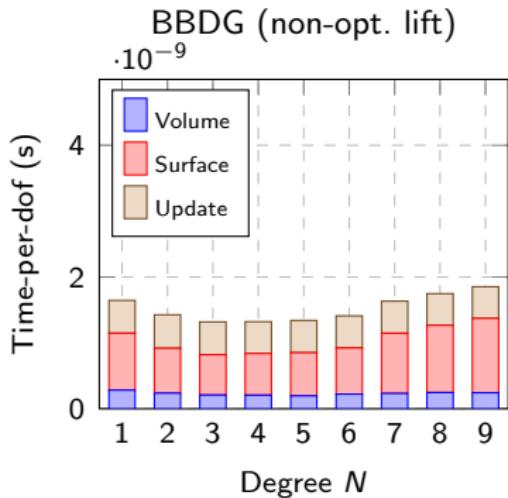
Performance comparisons of Bernstein-Bezier DG



Performance comparisons of Bernstein-Bezier DG

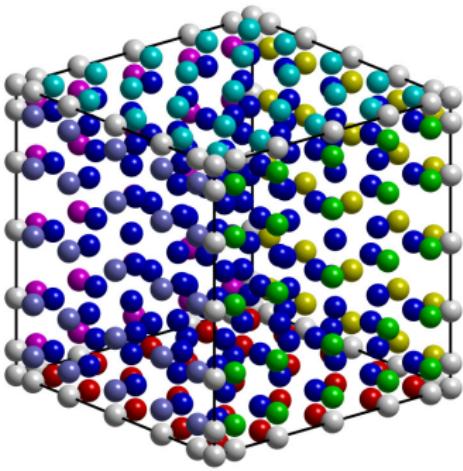


Preliminary comparisons: BBDG, SEM-DG on GPUs



- BBDG $1\text{--}1.75\times$ faster per dof than SEM-DG for $N \leq 10$.
- Unstructured hex meshes: $9(N + 1)^3$ geometric factors per element.
- Disclaimer: hexes are more accurate, need time-to-error studies!

Preliminary comparisons: BBDG, SEM-DG on GPUs



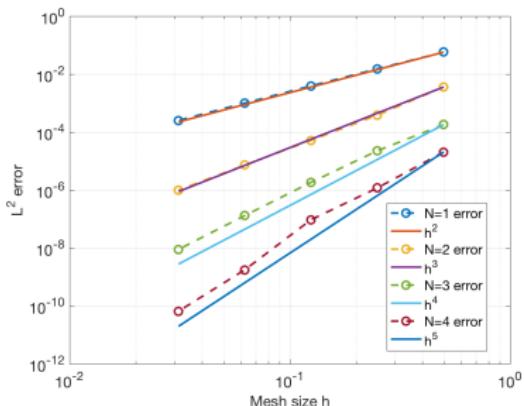
- BBDG $1\text{--}1.75\times$ faster per dof than SEM-DG for $N \leq 10$.
- Unstructured hex meshes: $9(N + 1)^3$ geometric factors per element.
- Disclaimer: hexes are more accurate, need time-to-error studies!

Effect of conservation on shock speeds

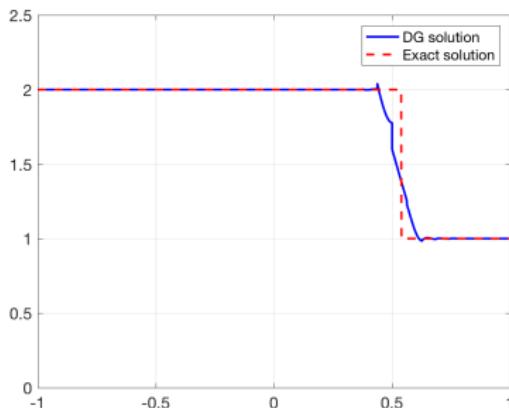
- Weighted Burgers' equation, $w(x)$ curves characteristic lines.

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

- WADG yields high order convergence, correct shock speed for both $w(x)$ smooth, discontinuous (within an element).



(n) Smooth solution



(o) Shock solution

Effect of conservation on shock speeds

- Weighted Burgers' equation, $w(x)$ curves characteristic lines.

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

- WADG yields high order convergence, correct shock speed for both $w(x)$ smooth, discontinuous (within an element).

Best guess: where and what is locally conserved matters;
non-conservation of *nonlinear flux* results in incorrect shock speeds.