Weight-adjusted Bernstein-Bezier DG method for wave propagation in heterogeneous media

Kaihang Guo, Jesse Chan

Department of Computational and Applied Mathematics
Rice University

North American High Order Methods Conference
June 2-5, 2019

First-order wave equations

- Acoustic wave equation:

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

- Elastic wave equation:

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

- Numerical scheme: high order, explicit time-stepping, parallelizable

High order DG methods for wave propagation

- Unstructured (tetrahedral) meshes for geometric flexibility.
- High order: low numerical dissipation and dispersion.
- High order approximations: more accurate per unknown.
- Explicit time stepping: high performance on many-core.

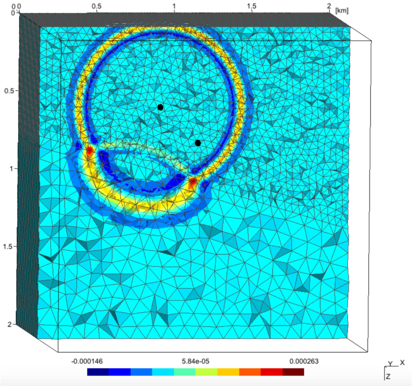
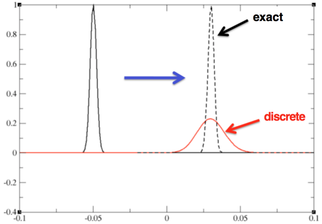


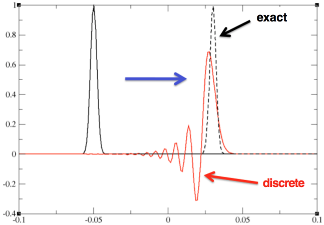
Figure courtesy of Axel Modave.

High order DG methods for wave propagation

- Unstructured (tetrahedral) meshes for geometric flexibility.
- High order: low numerical dissipation and dispersion.
- High order approximations: more accurate per unknown.
- Explicit time stepping: high performance on many-core.



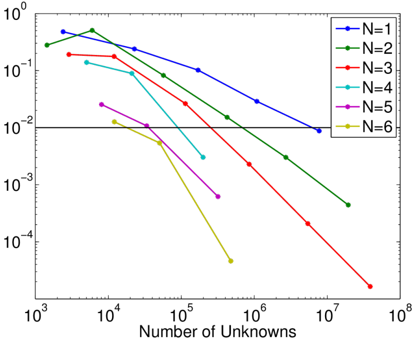
(a) Dissipative error



(b) Dispersive error

High order DG methods for wave propagation

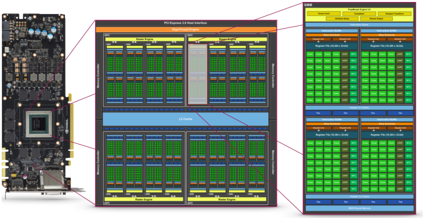
- Unstructured (tetrahedral) meshes for geometric flexibility.
- High order: low numerical dissipation and dispersion.
- High order approximations: more accurate per unknown.
- Explicit time stepping: high performance on many-core.



Max errors vs. dofs.

High order DG methods for wave propagation

- Unstructured (tetrahedral) meshes for geometric flexibility.
- High order: low numerical dissipation and dispersion.
- High order approximations: more accurate per unknown.
- Explicit time stepping: high performance on many-core.

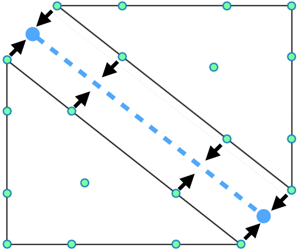


Graphics processing units (GPU).

Time-domain nodal DG methods

Assume $u(\boldsymbol{x}, t) = \sum \boldsymbol{u}_j \phi_j(\boldsymbol{x})$ on D^k

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



$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = \mathbf{D}_x \mathbf{u} + \sum_{\text{faces}} \mathbf{L}_f \left(\text{flux}\right).$$

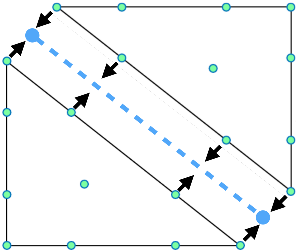
$$\boldsymbol{M}_{ij} = \int_{D^k} \phi_j(\boldsymbol{x}) \phi_i(\boldsymbol{x})$$

$$\mathbf{L}_f = \mathbf{M}^{-1} \mathbf{M}_f.$$

Time-domain nodal DG methods

Assume $u(\boldsymbol{x}, t) = \sum \boldsymbol{u}_j \phi_j(\boldsymbol{x})$ on D^k

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



$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = \underbrace{\mathbf{D}_x \mathbf{u}}_{\text{Volume kernel}} + \underbrace{\sum_{\text{faces}} \mathbf{L}_f \left(\text{flux}\right)}_{\text{Surface kernel}}.$$

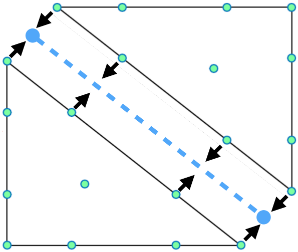
$$\boldsymbol{M}_{ij} = \int_{D^k} \phi_j(\boldsymbol{x}) \phi_i(\boldsymbol{x})$$

$$\mathbf{L}_f = \mathbf{M}^{-1} \mathbf{M}_f.$$

Time-domain nodal DG methods

Assume $u(\boldsymbol{x},t) = \sum \boldsymbol{u}_j \phi_j(\boldsymbol{x})$ on D^k

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



$$\underbrace{\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t}}_{\text{Update kernel}} = \underbrace{\mathbf{D}_x \mathbf{u}}_{\text{Volume kernel}} + \underbrace{\sum_{\text{faces}} \mathbf{L}_f \left(\text{flux}\right)}_{\text{Surface kernel}}.$$

$$\boldsymbol{M}_{ij} = \int_{D^k} \phi_j(\boldsymbol{x}) \phi_i(\boldsymbol{x})$$

$$\mathbf{L}_f = \mathbf{M}^{-1} \mathbf{M}_f.$$

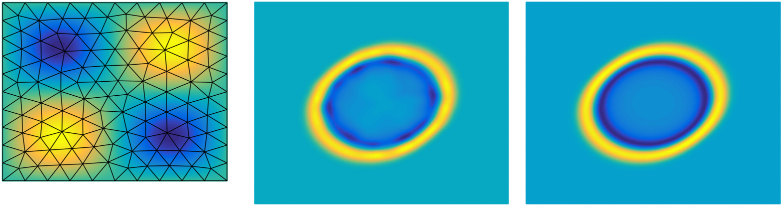
Outline

- Weight-adjusted DG (WADG): high order wavespeed
- Bernstein-Bézier DG (BBDG): piecewise constant wavespeed
- Bernstein-Bézier WADG (BBWADG)

Outline

- Weight-adjusted DG (WADG): high order wavespeed
- Bernstein-Bézier DG (BBDG): piecewise constant wavespeed
- Bernstein-Bézier WADG (BBWADG)

High order approximation of media and geometry



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

- Piecewise constant wavespeed c^2: efficient, but spurious reflections.

$$\frac{1}{c^2(\boldsymbol{x})}\frac{\partial p}{\partial t} + \nabla \cdot \boldsymbol{u} = 0, \qquad \frac{\partial \boldsymbol{u}}{\partial t} + \nabla p = 0.$$

- High order wavespeeds: weighted mass matrices. Stable, but requires pre-computation/storage of inverses or factorizations!

$$\boldsymbol{M}_{1/c^2}\frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t} = \boldsymbol{A}_h\boldsymbol{U}, \qquad \left(\boldsymbol{M}_{1/c^2}\right)_{ij} = \int_{D^k} \frac{1}{c^2(\boldsymbol{x})}\phi_j(\boldsymbol{x})\phi_i(\boldsymbol{x}).$$

Weight-adjusted mass matrix

- Weight-adjusted DG (WADG): energy stable approximation of $\boldsymbol{M}^k_{1/c^2}$

$$\boldsymbol{M}^k_{1/c^2}\frac{d\boldsymbol{p}}{dt} \approx \boldsymbol{M}^k\left(\boldsymbol{M}^k_{c^2}\right)^{-1}\boldsymbol{M}^k\frac{d\boldsymbol{p}}{dt} = \boldsymbol{A}_h\boldsymbol{U}$$

- Reuses implementation for piecewise constant wavespeed

$$\frac{d\boldsymbol{p}}{dt} = \underbrace{\left(\boldsymbol{M}^k\right)^{-1}\left(\boldsymbol{M}^k_{c^2}\right)}_{\text{modified update}} \underbrace{\left(\boldsymbol{M}^k\right)^{-1}\boldsymbol{A}_h\boldsymbol{U}}_{\text{RHS for } c=1}$$

- Modified update can be applied in a low storage manner using quadrature-based interpolation $\boldsymbol{V}_q$ and L^2 projection $\boldsymbol{P}_q$.

Chan, Hewett, Warburton. 2017. Weight-adjusted discontinuous Galerkin methods: wave propagation in heterogeneous media.

Quadrature-based operators

- Using quadrature rule

$$\left(\boldsymbol{M}^k_{c^2}\right)_{ij} = \int_{D^k} c^2(\boldsymbol{x})\phi_i^k(\boldsymbol{x})\phi_j^k(\boldsymbol{x})d\boldsymbol{x} = J^k \sum_{n=1}^{N_q} c^2(\boldsymbol{x}_n^q)\phi_i(\hat{\boldsymbol{x}}_n^q)\phi_j(\hat{\boldsymbol{x}}_n^q)\boldsymbol{w}_n$$

where $\boldsymbol{x}^q, \hat{\boldsymbol{x}}^q$ denote quadrature points on D^k and $\hat{D}$, respectively.

- Writing $\boldsymbol{M}^k_{c^2}$ into matrix form

$$\boldsymbol{M}^k_{c^2} = J^k \boldsymbol{V}_q^T \operatorname{diag}(\boldsymbol{w}) \operatorname{diag}\left(c^2\right) \boldsymbol{V}_q$$

where

$$(\boldsymbol{V}_q)_{ij} = \phi_j(\hat{\boldsymbol{x}}_i^q)$$

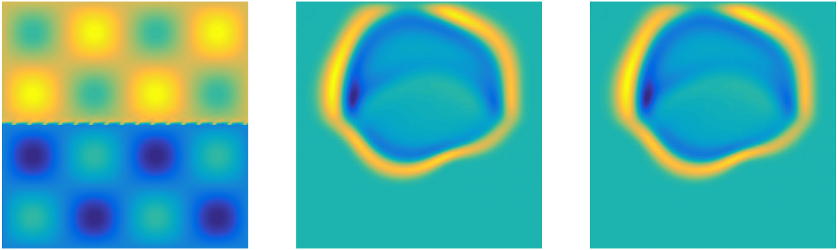
Quadrature-based operators

- The modified update can be written as

$$\left(\boldsymbol{M}^k\right)^{-1}\left(\boldsymbol{M}^k_{c^2}\right) = \underbrace{\boldsymbol{M}^{-1}\boldsymbol{V}_q^T \operatorname{diag}(\boldsymbol{w})}_{\boldsymbol{P}_q} \operatorname{diag}\left(c^2\right) \boldsymbol{V}_q$$

- $\boldsymbol{V}_q$: evaluates function values at quadrature points.
- $\boldsymbol{P}_q$: projects a function onto a polynomial space in L^2 sense.

Wave simulations in heterogeneous media



(a) $c^2(x, y)$ (b) Standard DG (c) Weight-adjusted DG

- L^2 convergence between optimal $O(h^{N+1})$, provable $O(h^{N+1/2})$.
- Difference between standard DG and WADG is $O(h^{N+2})$.

Chan, Hewett, Warburton. 2017. Weight-adjusted DG methods: wave propagation in heterogeneous media.

WADG implementation

- In WADG

$$\frac{d\boldsymbol{p}}{dt} = \underbrace{\boldsymbol{P}_q \operatorname{diag}\left(c^2\right) \boldsymbol{V}_q}_{\text{modified update}} \underbrace{\left(\boldsymbol{M}^k\right)^{-1} \boldsymbol{A}_h \boldsymbol{U}}_{\text{RHS for } c=1}$$

- RHS for $c = 1$ produces a polynomial u.
- $\boldsymbol{V}_q$ evaluates values of u at quadrature points.
- Applying $\operatorname{diag}\left(c^2\right)$ to u gives the product $c^2 u$.
- $\boldsymbol{P}_q$ projects $c^2 u$ onto a polynomial space of degree N.

WADG implementation

- In WADG

$$\frac{d\boldsymbol{p}}{dt} = \underbrace{\boldsymbol{P}_q \operatorname{diag}\left(c^2\right) \boldsymbol{V}_q}_{\text{modified update}} \underbrace{\left(\boldsymbol{M}^k\right)^{-1} \boldsymbol{A}_h \boldsymbol{U}}_{\text{RHS for } c=1}$$

- RHS for $c = 1$ produces a polynomial u.
- $\boldsymbol{V}_q$ evaluates values of u at quadrature points.
- Applying $\operatorname{diag}\left(c^2\right)$ to u gives the product $c^2 u$.
- $\boldsymbol{P}_q$ projects $c^2 u$ onto a polynomial space of degree N.

WADG implementation

- In WADG

$$\frac{d\boldsymbol{p}}{dt} = \underbrace{\boldsymbol{P}_q \operatorname{diag}\left(c^2\right) \boldsymbol{V}_q}_{\text{modified update}} \underbrace{\left(\boldsymbol{M}^k\right)^{-1} \boldsymbol{A}_h \boldsymbol{U}}_{\text{RHS for } c=1}$$

- RHS for $c = 1$ produces a polynomial u.
- $\boldsymbol{V}_q$ evaluates values of u at quadrature points.
- Applying $\operatorname{diag}\left(c^2\right)$ to u gives the product $c^2 u$.
- $\boldsymbol{P}_q$ projects $c^2 u$ onto a polynomial space of degree N.

WADG implementation

- In WADG

$$\frac{d\boldsymbol{p}}{dt} = \underbrace{\boldsymbol{P}_q \operatorname{diag}\left(c^2\right) \boldsymbol{V}_q}_{\text{modified update}} \underbrace{\left(\boldsymbol{M}^k\right)^{-1} \boldsymbol{A}_h \boldsymbol{U}}_{\text{RHS for } c=1}$$

- RHS for $c = 1$ produces a polynomial u.
- $\boldsymbol{V}_q$ evaluates values of u at quadrature points.
- Applying $\operatorname{diag}\left(c^2\right)$ to u gives the product $c^2 u$.
- $\boldsymbol{P}_q$ projects $c^2 u$ onto a polynomial space of degree N.

WADG implementation

- In WADG

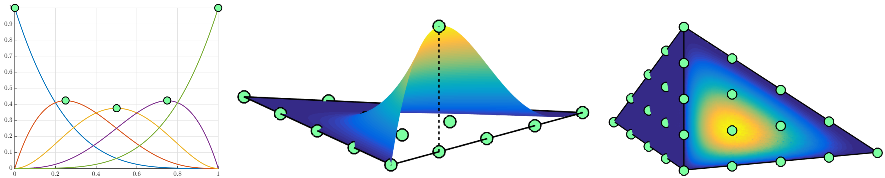
$$\frac{d\boldsymbol{p}}{dt} = \underbrace{\boldsymbol{P}_q \operatorname{diag}\left(c^2\right) \boldsymbol{V}_q}_{\text{modified update}} \underbrace{\left(\boldsymbol{M}^k\right)^{-1} \boldsymbol{A}_h \boldsymbol{U}}_{\text{RHS for } c=1}$$

- RHS for $c = 1$ produces a polynomial u.
- $\boldsymbol{V}_q$ evaluates values of u at quadrature points.
- Applying $\operatorname{diag}\left(c^2\right)$ to u gives the product $c^2 u$.
- $\boldsymbol{P}_q$ projects $c^2 u$ onto a polynomial space of degree N.

Outline

- Weight-adjusted DG (WADG): high order wavespeed
- Bernstein-Bézier DG (BBDG): piecewise constant wavespeed
- Bernstein-Bézier WADG (BBWADG)

Bernstein polynomial bases on simplices



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

- Bernstein polynomials are associated with vertex, edge, face, and interior equispaced nodes.
- Jumps across interfaces can be computed similarly to nodal bases (reuse nodal DG framework).

Bernstein polynomial

- On the reference tetrahedron, the barycentric coordinates are

$$\lambda_0 = -\frac{(1+r+s+t)}{2},\ \lambda_1 = \frac{(1+r)}{2},\ \lambda_2 = \frac{(1+s)}{2},\ \lambda_3 = \frac{(1+t)}{2}.$$

- The Nth degree Bernstein basis is defined as

$$B_{\boldsymbol{\alpha}}^N = C_{\boldsymbol{\alpha}}^N \lambda_0^{\alpha_0}\lambda_1^{\alpha_1}\lambda_2^{\alpha_2}\lambda_3^{\alpha_3}, \qquad C_{\boldsymbol{\alpha}}^N = \frac{N!}{\alpha_0!\alpha_1!\alpha_2!\alpha_3!},$$

where $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, \alpha_2, \alpha_3)$ and $\alpha_0 + \alpha_1 + \alpha_2 + \alpha_3 = N$.

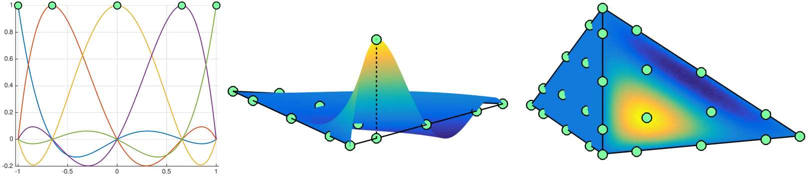
- Simple degree elevation of Bernstein polynomials

$$B_{\boldsymbol{\alpha}}^{N-1} = \sum_{j=0}^{d} \frac{\alpha_j + 1}{N} B_{\boldsymbol{\alpha}+\boldsymbol{e}_j}^N$$

leads to sparse one-degree elevation operators.

BBDG: Bernstein-Bezier DG methods

- Nodal DG: $O(N^6)$ cost in 3D vs $O(N^3)$ degrees of freedom.
- Switch to Bernstein basis: sparse and structured matrices.
- Optimal $O(N^3)$ application of differentiation and lifting matrices.

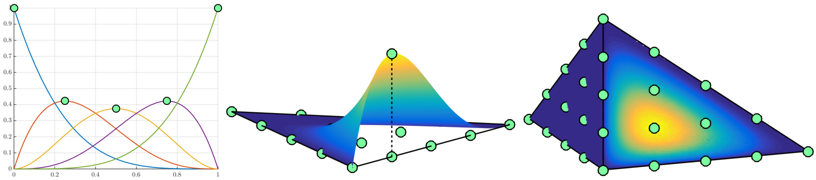


Nodal bases in one, two, and three dimensions.

Chan, Warburton 2015. GPU-accelerated Bernstein-Bezier discontinuous Galerkin methods for wave propagation (SISC).

BBDG: Bernstein-Bezier DG methods

- Nodal DG: $O(N^6)$ cost in 3D vs $O(N^3)$ degrees of freedom.
- Switch to Bernstein basis: sparse and structured matrices.
- Optimal $O(N^3)$ application of differentiation and lifting matrices.

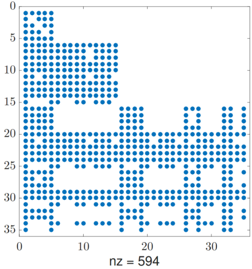


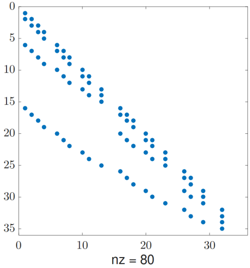
Bernstein bases in one, two, and three dimensions.

Chan, Warburton 2015. GPU-accelerated Bernstein-Bezier discontinuous Galerkin methods for wave propagation (SISC).

BBDG: Bernstein-Bezier DG methods

- Nodal DG: $O(N^6)$ cost in 3D vs $O(N^3)$ degrees of freedom.
- Switch to Bernstein basis: sparse and structured matrices.
- Optimal $O(N^3)$ application of differentiation and lifting matrices.



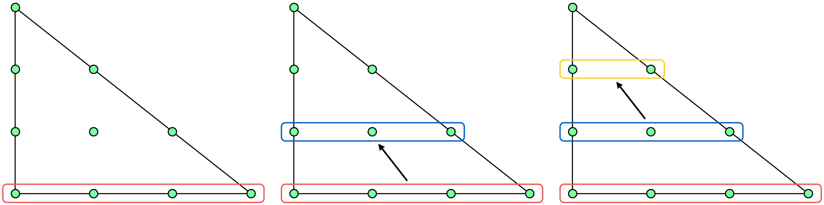


Sparse Bernstein differentiation matrices for the reference tetrahedron.

Chan, Warburton 2015. GPU-accelerated Bernstein-Bezier discontinuous Galerkin methods for wave propagation (SISC).

BBDG: Bernstein-Bezier DG methods

- Nodal DG: $O(N^6)$ cost in 3D vs $O(N^3)$ degrees of freedom.
- Switch to Bernstein basis: sparse and structured matrices.
- Optimal $O(N^3)$ application of differentiation and lifting matrices.

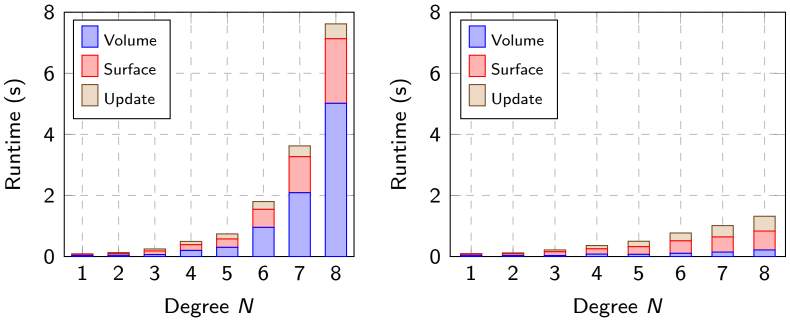


Optimal $O(N^3)$ complexity "slice-by-slice" application of Bernstein lift.

Chan, Warburton 2015. GPU-accelerated Bernstein-Bezier discontinuous Galerkin methods for wave propagation (SISC).

BBDG vs Nodal DG

Numerical experiments are implemented on a mesh with 61276 elements



(a) Nodal DG runtimes (b) BBDG runtimes

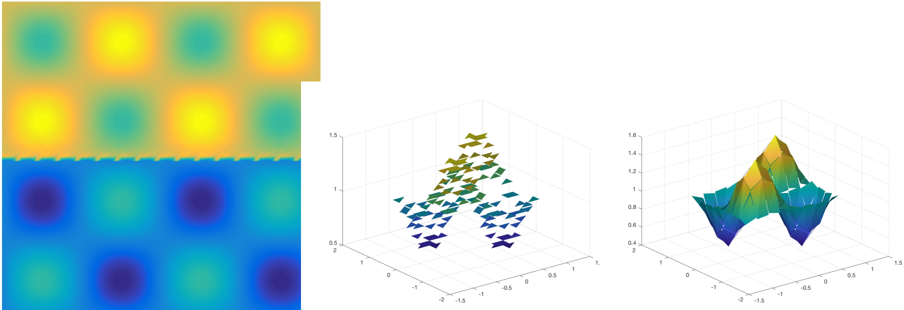
We observe BBDG achieves $\approx 6\times$ speedup at high orders ($N = 8$).

Chan, Warburton. 2017. GPU-accelerated Bernstein-Bézier discontinuous Galerkin methods for wave problems.

Outline

- Weight-adjusted DG (WADG): high order wavespeed
- Bernstein-Bézier DG (BBDG): piecewise constant wavespeed
- Bernstein-Bézier WADG (BBWADG)

BBWADG: polynomial multiplication and projection



(a) Exact c^2 (b) $M = 0$ approximation (c) $M = 1$ approximation

- BBWADG reuses the volume and surface kernels from BBDG
- Instead of using quadrature-based L^2 projection, BBWADG approximates $c^2(\boldsymbol{x})$ with degree M polynomial, use fast Bernstein algorithms for polynomial multiplication and projection.

Bernstein polynomial multiplication

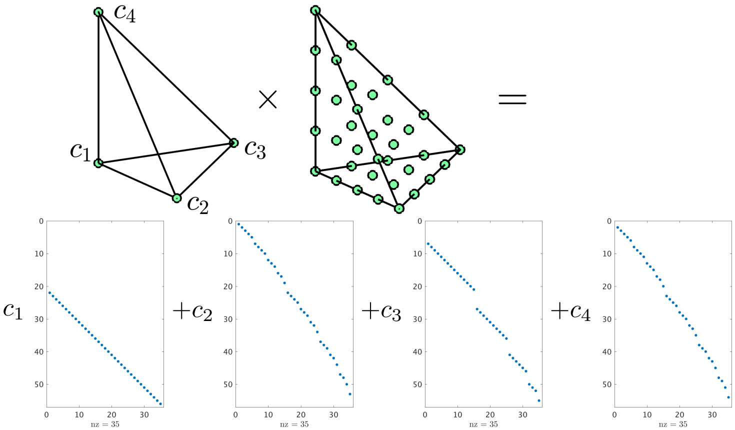
- Assume $h(\boldsymbol{x}) = f(\boldsymbol{x})g(\boldsymbol{x})$, where f, g are Bernstein polynomials

$$f = c_1\lambda_0 + c_2\lambda_1 + c_3\lambda_2 + c_4\lambda_3, \qquad g = \sum_{|\boldsymbol{\alpha}|\le N} d_{\boldsymbol{\alpha}} B_{\boldsymbol{\alpha}}^N$$

- We can split the multiplication into $d+1$ parts. For example, the first part is

$$c_1 \sum_{|\boldsymbol{\alpha}|\le N} d_{\boldsymbol{\alpha}} \left(B_{\boldsymbol{\alpha}}^N \lambda_0 \right) = c_1 \sum_{|\boldsymbol{\alpha}|\le N} d_{\boldsymbol{\alpha}} \frac{\alpha_j + 1}{N+1} B_{\boldsymbol{\alpha}+\boldsymbol{e}_0}^{N+1}$$

Bernstein polynomial multiplication



Bernstein polynomial multiplication: for fixed M, $O(N^3)$ complexity.

Fast Bernstein polynomial projection

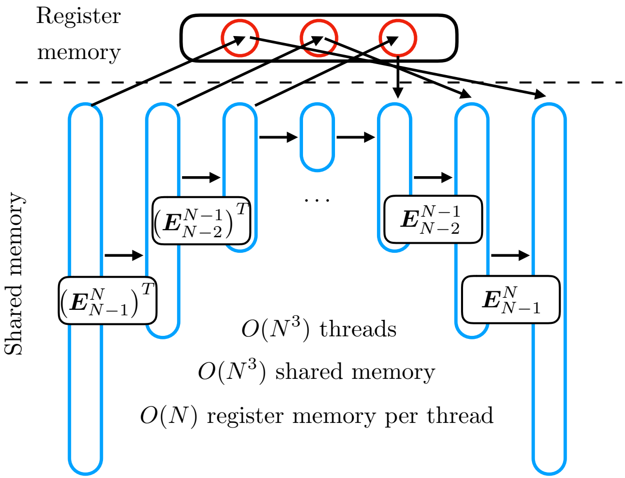
- Given $c^2(\boldsymbol{x})u(\boldsymbol{x})$ as a degree $(N+M)$ polynomial, apply L^2 projection matrix $\boldsymbol{P}_N^{N+M}$ to reduce to degree N.
- Polynomial L^2 projection matrix $\boldsymbol{P}_N^{N+M}$ under Bernstein basis:

$$\boldsymbol{P}_N^{N+M} = \underbrace{\sum_{j=0}^{N} c_j \boldsymbol{E}_{N-j}^{N} \left(\boldsymbol{E}_{N-j}^{N}\right)^T}_{\tilde{\boldsymbol{P}}_N} \left(\boldsymbol{E}_N^{N+M}\right)^T$$

- "Telescoping" form of $\tilde{\boldsymbol{P}}_N$: $O(N^4)$ complexity, more GPU-friendly.

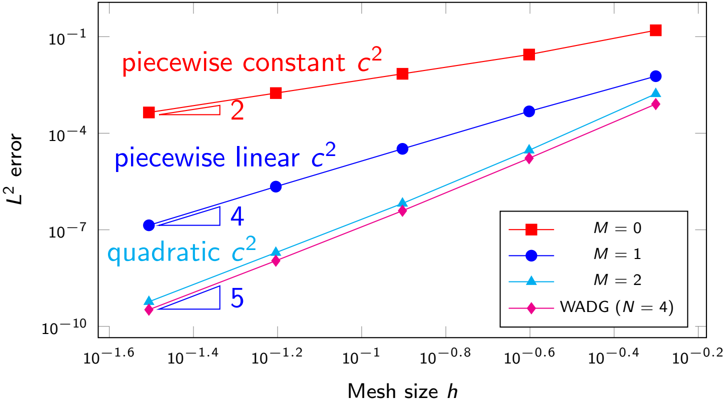
$$\left(c_0 \boldsymbol{I} + \boldsymbol{E}_{N-1}^{N}\left(c_1 \boldsymbol{I} + \boldsymbol{E}_{N-2}^{N-1}\left(c_2 \boldsymbol{I} + \cdots\right)\left(\boldsymbol{E}_{N-2}^{N-1}\right)^T\right)\left(\boldsymbol{E}_{N-1}^{N}\right)^T\right)$$

Illustration of GPU algorithm for $\tilde{P}_N$



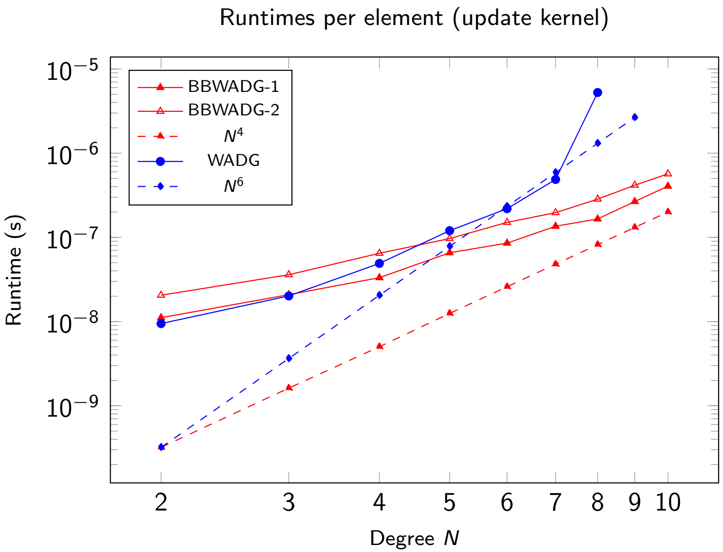
$$\left(c_0 \boldsymbol{I} + \boldsymbol{E}_{N-1}^{N}\left(c_1 \boldsymbol{I} + \boldsymbol{E}_{N-2}^{N-1}\left(c_2 \boldsymbol{I} + \cdots\right)\left(\boldsymbol{E}_{N-2}^{N-1}\right)^T\right)\left(\boldsymbol{E}_{N-1}^{N}\right)^T\right)$$

BBWADG: approximating c^2 and accuracy

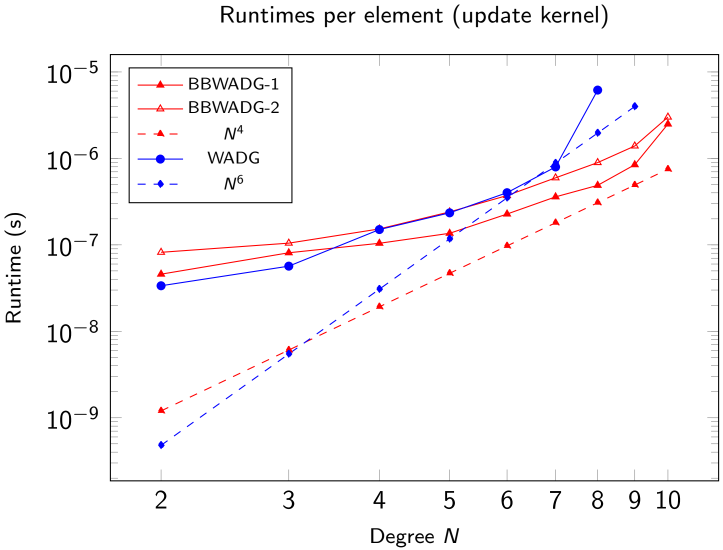


Approximating smooth $c^2(\boldsymbol{x})$ using L^2 projection:
$O(h^2)$ for $M = 0$, $O(h^4)$ for $M = 1$, $O(h^{M+3})$ for $0 < M \leq N - 2$.

BBWADG vs WADG (acoustic)



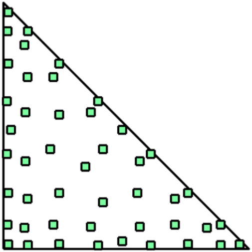
BBWADG vs WADG (elastic)



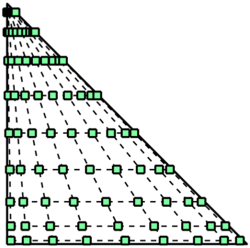
BBWADG vs WADG (acoustic)

	$N = 3$	$N = 4$	$N = 5$	$N = 6$	$N = 7$	$N = 8$
WADG	2.02e-8	4.91e-8	1.20e-7	2.19e-7	4.87e-7	5.25e-6
BBWADG-1	2.09e-8	3.32e-8	6.56e-8	8.54e-8	1.35e-7	1.65e-7
Speedup	0.9665	1.4789	1.8292	2.5644	3.6074	31.8182

For $N \geq 8$, WADG becomes much more expensive because of the quadrature points construction.



(d) $N = 7$ quadrature



(e) $N = 8$ quadrature

Summary and acknowledgements

- Weight-adjusted DG: stability and efficiency for heterogeneous media.
- BBWADG: improved complexity for approximate wavespeeds.
- This work is supported by the National Science Foundation under DMS-1712639 and DMS-1719818.

Thank you! Questions?

Guo, Chan. 2018. Bernstein-Bezier weight-adjusted discontinuous Galerkin methods for wave propagation.

Chan, Hewett, Warburton. 2016. Weight-adjusted DG methods: wave propagation in heterogeneous media (SISC).

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

Chan, Warburton 2015. GPU-accelerated Bernstein-Bezier discontinuous Galerkin methods for wave propagation (SISC).