Time-domain Bernstein-Bézier DG methods on simplices

Jesse Chan, T. Warburton

[1]Department of Computational and Applied Mathematics
Rice University

27th Biennial Numerical analysis conference
University of Strathclyde
June 27, 2017

Outline and acknowledgments

Collaborators and contributors:

- T. Warburton (Virginia Tech)
- Russell J. Hewett (TOTAL E&P Research and Technology USA)

High order DG methods for wave propagation

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

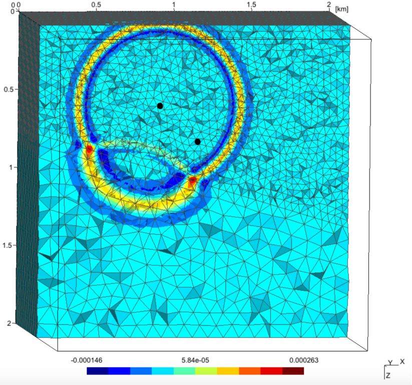
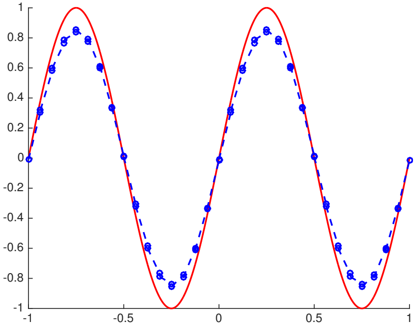


Figure courtesy of Axel Modave.

High order DG methods for wave propagation

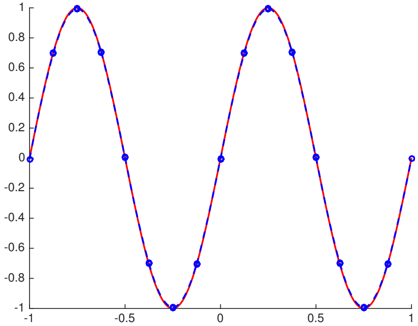
- Unstructured (tetrahedral) meshes for geometric flexibility.
- Low numerical dissipation and dispersion.
- High order approximations: more accurate per unknown.
- High performance on many-core (explicit time stepping).



Fine linear approximation.

High order DG methods for wave propagation

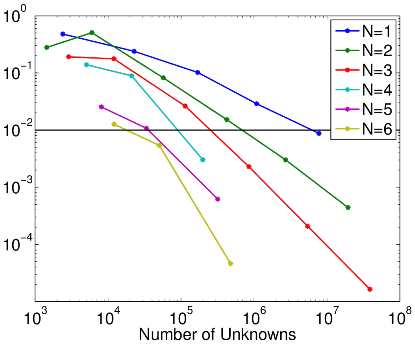
- Unstructured (tetrahedral) meshes for geometric flexibility.
- Low numerical dissipation and dispersion.
- High order approximations: more accurate per unknown.
- High performance on many-core (explicit time stepping).



Coarse quadratic approximation.

High order DG methods for wave propagation

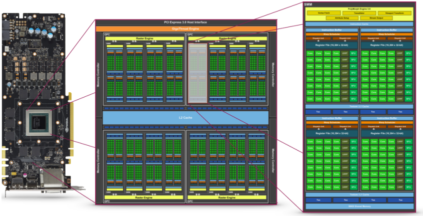
- Unstructured (tetrahedral) meshes for geometric flexibility.
- Low numerical dissipation and dispersion.
- High order approximations: more accurate per unknown.
- High performance on many-core (explicit time stepping).



Max errors vs. dofs.

High order DG methods for wave propagation

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

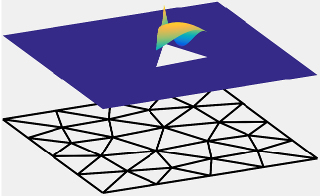


A graphics processing unit (GPU).

Discontinuous Galerkin methods

Discontinuous Galerkin (DG) methods:

- Piecewise polynomial approximation.
- Weak continuity across faces.



- Continuous PDE (for illustration: constant advection)

$$\frac{\partial u}{\partial t} = \frac{\partial u}{\partial x}$$

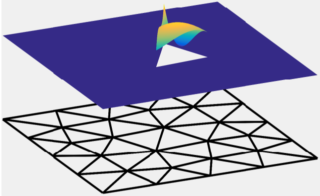
- DG local weak form over D_k with numerical flux $\boldsymbol{f}^*$.

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

Discontinuous Galerkin methods

Discontinuous Galerkin (DG) methods:

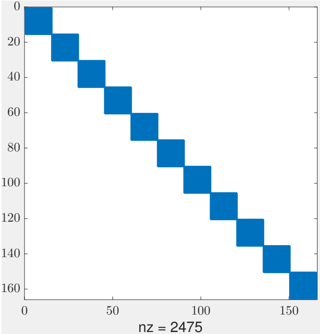
- Piecewise polynomial approximation.
- Weak continuity across faces.



DG yields system of ODEs

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

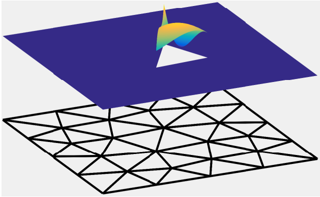
DG mass matrix decouples across elements, inter-element coupling only through $\mathbf{A}$.



Discontinuous Galerkin methods

Discontinuous Galerkin (DG) methods:

- Piecewise polynomial approximation.
- Weak continuity across faces.



- Matrix-free evaluation of $\mathbf{M}^{-1}\mathbf{A}$.
- Local differentiation and lifting matrices $\mathbf{D}_x$ and $\mathbf{L}_f = \mathbf{M}^{-1}\mathbf{M}_f$.
- Assume (for now) piecewise constant coefficients and affine mappings.

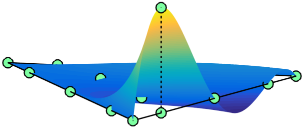
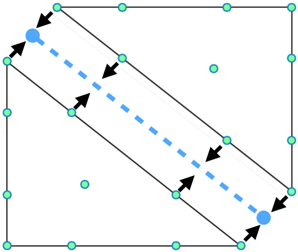


Figure: Nodal bases simplify flux computations.

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).



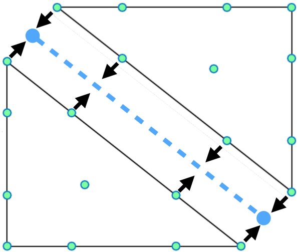
$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = \mathbf{D}_x\mathbf{u} + \sum_{\text{faces}} \mathbf{L}_f\left(\text{flux}\right), \qquad \mathbf{L}_f = \mathbf{M}^{-1}\mathbf{M}_f.$$

Well-suited to GPUs computing (**significant** speedups over CPUs)!

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).



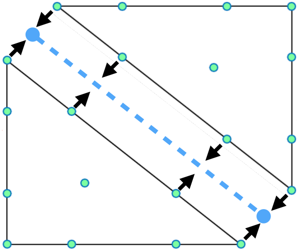
$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = \underbrace{\mathbf{D}_x \mathbf{u}}_{\text{Volume kernel}} + \underbrace{\sum_{\text{faces}} \mathbf{L}_f\,(\text{flux})}_{\text{Surface kernel}}, \qquad \mathbf{L}_f = \mathbf{M}^{-1}\mathbf{M}_f.$$

Well-suited to GPUs computing (**significant** speedups over CPUs)!

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).



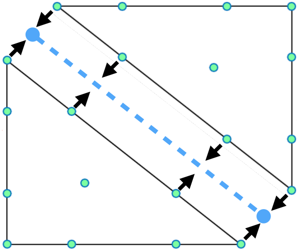
$$\underbrace{\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t}}_{\text{Update kernel}} = \underbrace{\mathbf{D}_x\mathbf{u}}_{\text{Volume kernel}} + \underbrace{\sum_{\text{faces}} \mathbf{L}_f\,(\text{flux})}_{\text{Surface kernel}}, \qquad \mathbf{L}_f = \mathbf{M}^{-1}\mathbf{M}_f.$$

Well-suited to GPUs computing (**significant** speedups over CPUs)!

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).



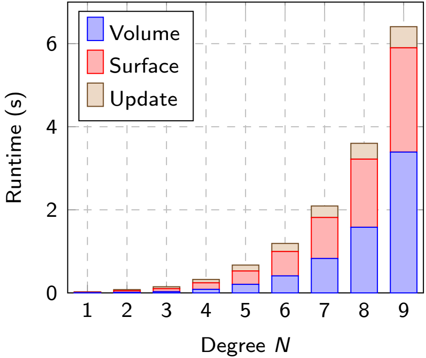
$$\underbrace{\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t}}_{\text{Update kernel}} = \underbrace{\mathbf{D}_x \mathbf{u}}_{\text{Volume kernel}} + \underbrace{\sum_{\text{faces}} \mathbf{L}_f \, (\text{flux})}_{\text{Surface kernel}}, \qquad \mathbf{L}_f = \mathbf{M}^{-1}\mathbf{M}_f.$$

Well-suited to GPUs computing (**significant** speedups over CPUs)!

Computational costs at high orders of approximation

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

Nodal DG runtimes



DG runtimes for 50 timesteps, 98304 elements.

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

Spectral element methods

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

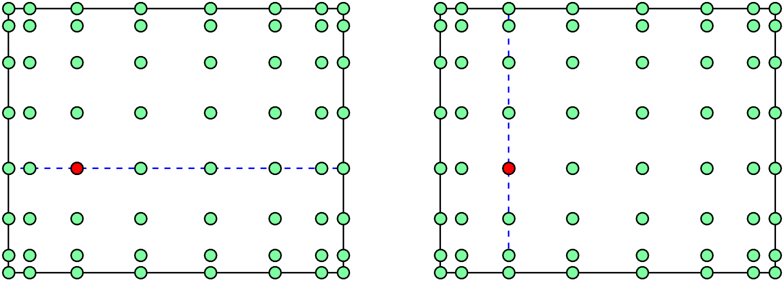


Figure: Spectral element stencils for $N = 7$ (orders $N > 10$ not uncommon!).

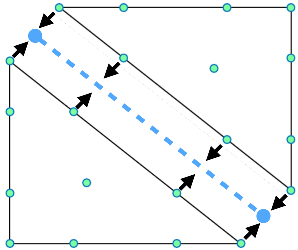
Fischer, Ronquist 1994. Spectral element methods for large scale parallel Navier-Stokes calculations.

Shepherd and Johnson 2008. Hexahedral mesh generation constraints.

High order nodal DG on tetrahedral meshes

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

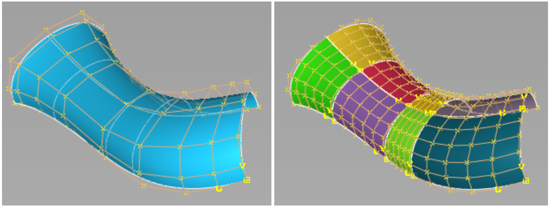
- Nodal bases: reduce the cost of computing numerical fluxes.
- No clear tetrahedral equivalent to spectral differentiation, lift matrices.
- $O(N^3)$ unknowns in 3D; $O(N^6)$ costs for applying **dense** matrices.



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

Bernstein-Bézier bases for finite element methods

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



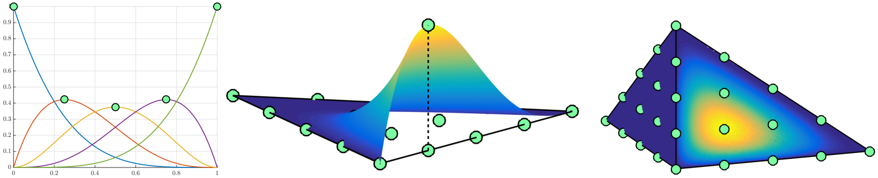
- Recent developments: optimal complexity assembly of finite element matrices, sum factorization (reduced complexity quadrature).
- This work: adapt Bernstein-Bézier for time-domain DG methods.

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

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

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

Bernstein-Bézier polynomial bases on simplices



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

- Simple expression in 1D

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

- 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-Bézier 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

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

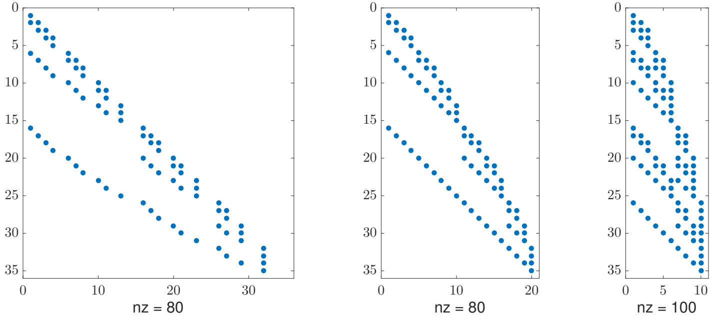
- Combine to get expansion of Bernstein derivatives

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

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

Bernstein-Bézier derivative and degree elevation in 3D

- Bernstein-Bézier 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 $\mathbf{E}_{N-2}^{N} = \mathbf{E}_{N-1}^{N}\mathbf{E}_{N-2}^{N-1}$.



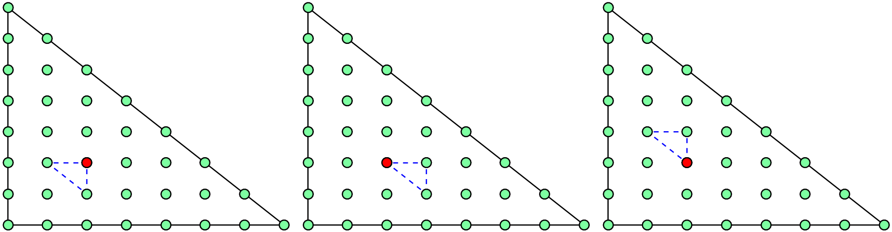
(a) Derivative matrix w.r.t. first barycentric coordinate.

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

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

Stencils for Bernstein-Bézier derivative matrices

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



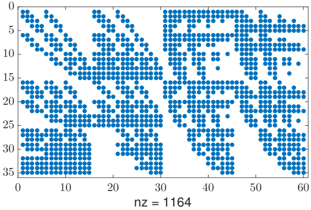
(d) Stencil for $\boldsymbol{D}_{\lambda_0}$ (e) Stencil for $\boldsymbol{D}_{\lambda_1}$ (f) Stencil for $\boldsymbol{D}_{\lambda_2}$

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

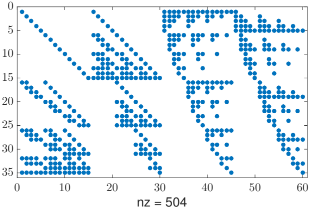
Factorization of the Bernstein lift operator

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

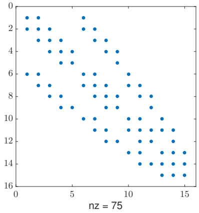
$$\mathbf{L} = \mathbf{E}_L \begin{pmatrix} \mathbf{L}_0 & & & \\ & \mathbf{L}_0 & & \\ & & \mathbf{L}_0 & \\ & & & \mathbf{L}_0 \end{pmatrix}.$$



(a) Lift matrix $\mathbf{L}$



(b) Lift reduction $\mathbf{E}_L$



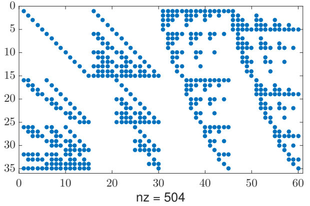
(c) $\mathbf{L}_0$

Chan, Warburton 2016. GPU-accelerated Bernstein-Bézier DG methods for wave problems.

Factorization of the Bernstein lift operator

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

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



$\mathbf{E}_L = \left[\ \mathbf{E}_L^1 \ \middle|\ \ldots \ \middle|\ \mathbf{E}_L^4\ \right]$ (4 faces).

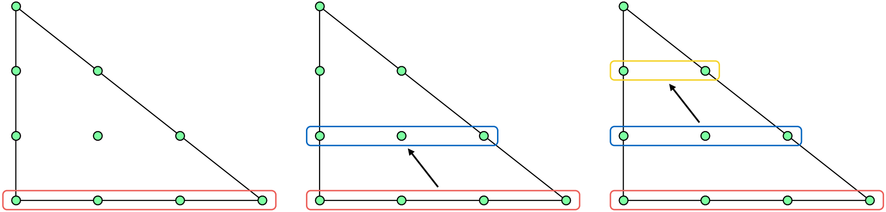
$$\mathbf{E}_L^1 = \begin{bmatrix} \mathbf{I} \\ \ell_1 \left(\mathbf{E}_{N-1}^N\right)^T \\ \vdots \\ \ell_N \left(\mathbf{E}_0^N\right)^T \end{bmatrix}.$$

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

Chan, Warburton 2016. GPU-accelerated Bernstein-Bézier DG methods for wave problems.

Bernstein-Bézier lift matrix: optimal complexity application

- $\boldsymbol{L}$ "lifts" numerical fluxes from faces to volume.
- Apply $\boldsymbol{L}_0$ to face flux, extend to each "layer" of the simplex.



(a) Apply $\boldsymbol{L}_0$ 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.

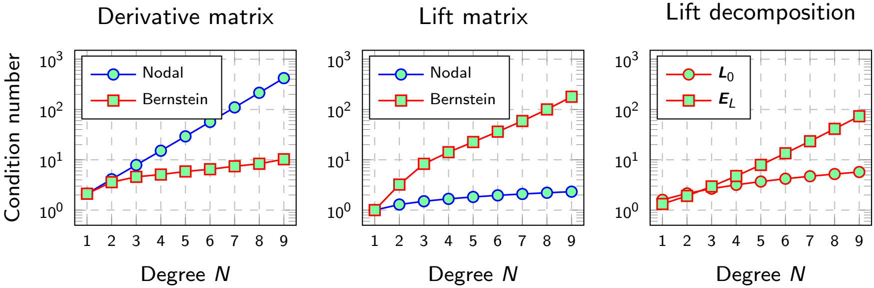
For $N < 6$, currently more efficient to treat $\boldsymbol{E}_L$ as a sparse matrix — irregular data accesses with optimal $O(N^d)$ approach.

Numerical stability of Bernstein-Bézier DG

- "Condition number" of Bernstein differentiation and lift matrices comparable to that of nodal bases.

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

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



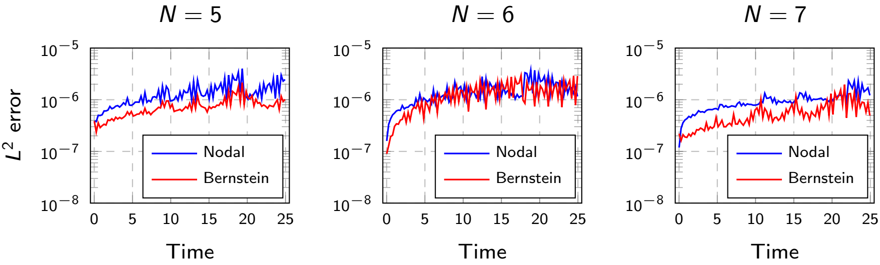
Condition numbers of matrices for nodal and Bernstein-Bézier bases.

Numerical stability of Bernstein-Bézier DG

- "Condition number" of Bernstein differentiation and lift matrices comparable to that of nodal bases.

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

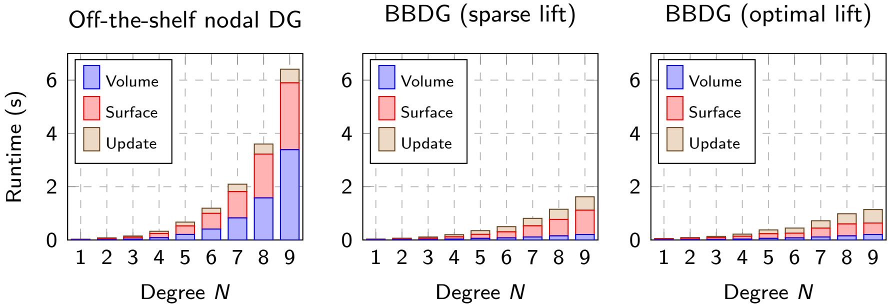
- Comparable long-time growth of (single precision) numerical error.



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

GPU runtime comparisons of BBDG and nodal DG

Bernstein-Bézier DG achieves $\approx 2\times$ speedup at moderate orders, and up to $\approx 6\times$ speedup at high orders (for acoustics).

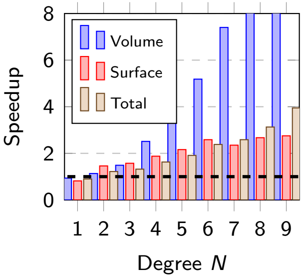


$$\underbrace{\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t}}_{\text{Update kernel}} = \underbrace{\mathbf{D}_x\mathbf{u}}_{\text{Volume kernel}} + \underbrace{\sum_{\text{faces}} \mathbf{L}_f\,(\text{flux})}_{\text{Surface kernel}}, \qquad \mathbf{L}_f = \mathbf{M}^{-1}\mathbf{M}_f.$$

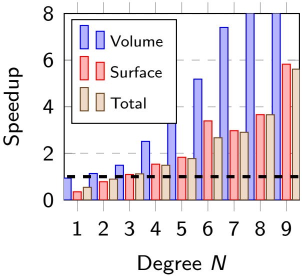
GPU runtime comparisons of BBDG and nodal DG

Bernstein-Bézier DG achieves $\approx 2\times$ speedup at moderate orders, and up to $\approx 6\times$ speedup at high orders (for acoustics).

BB speedup (sparse lift) over nodal



BB speedup (optimal lift) over nodal



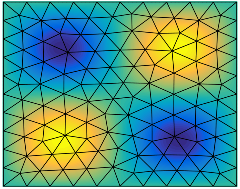
$$\underbrace{\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t}}_{\text{Update kernel}} = \underbrace{\mathbf{D}_x\mathbf{u}}_{\text{Volume kernel}} + \underbrace{\sum_{\text{faces}} \mathbf{L}_f\,(\text{flux})}_{\text{Surface kernel}}, \qquad \mathbf{L}_f = \mathbf{M}^{-1}\mathbf{M}_f.$$

Extensions: high order models of heterogeneous media

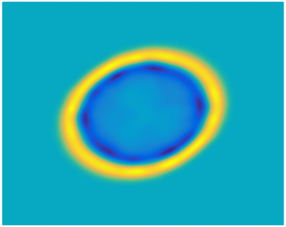
- Acoustic wave equation in heterogeneous media

$$\frac{1}{c^2(\boldsymbol{x})}\frac{\partial^2 p}{\partial t^2} - \Delta p = 0.$$

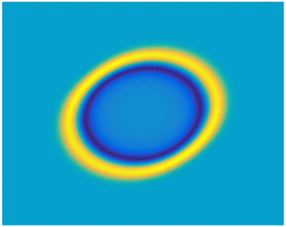
- Piecewise constant $c^2(\boldsymbol{x})$ efficient, but generates spurious reflections.
- Goal: high order $c^2(\boldsymbol{x})$, stability, low computational complexity.



(l) Mesh and exact c^2



(m) Piecewise constant c^2



(n) High order c^2

Weighted mass matrices and weight-adjusted DG

- Weighted mass matrix: high order accurate and energy stable, but high storage costs, $O(N^6)$ complexity to apply $\boldsymbol{M}_w^{-1}$.

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{M}_w\boldsymbol{u} = \text{right hand side}, \qquad w = 1/c^2.$$

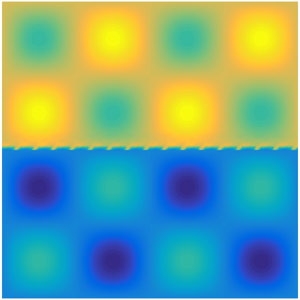
- Weight-adjusted DG (WADG): energy stable, low storage approximation of weighted mass matrix

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

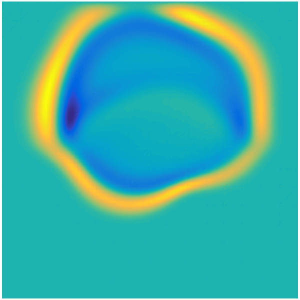
- Bypass inverse of weighted matrix $(\boldsymbol{M}_w)^{-1}$

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

Acoustic wave equation: heterogeneous media



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



(b) Standard DG

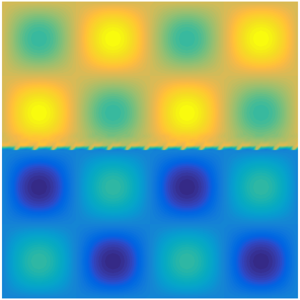
- L^2 convergence between optimal $O(h^{N+1})$, provable $O(h^{N+1/2})$.
- Extensions to curved elements, matrix weights (elastodynamics).

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

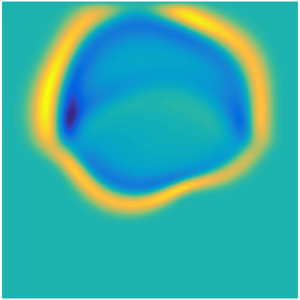
Chan, Hewett, Warburton. 2016. Weight-adjusted DG methods: curvilinear meshes.

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

Acoustic wave equation: heterogeneous media



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



(b) Weighted-adjusted DG

- L^2 convergence between optimal $O(h^{N+1})$, provable $O(h^{N+1/2})$.
- Extensions to curved elements, matrix weights (elastodynamics).

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

Chan, Hewett, Warburton. 2016. Weight-adjusted DG methods: curvilinear meshes.

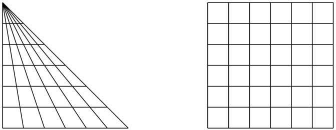
Chan 2017. Weight-adjusted DG methods: matrix-valued weights and elastic wave prop. in heterogeneous media.

WADG: low-complexity implementations

- Low storage, matrix-free application of $\left(\boldsymbol{M}_w^{-1}\boldsymbol{M}\right)^{-1} = \boldsymbol{M}^{-1}\boldsymbol{M}_w$.

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

- $O(N^4)$ cost in 3D: sum factorization for $\boldsymbol{V}_q$, block LDL for $\widehat{\boldsymbol{M}}^{-1}$.



- Current work: for fixed approximations of $w(\boldsymbol{x})$, optimal complexity WADG using polynomial multiplication and truncation.

Kirby 2017. Fast inversion of the simplicial Bernstein mass matrix.

Kirby and Thinh 2012. Fast simplicial quadrature-based finite element operators using Bernstein polynomials.

Summary and acknowledgements

- Optimal complexity RHS evaluation for time-domain DG.
- Bernstein-Bézier sparsity: efficiency at high orders on GPUs.

Thanks to NSF and 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, Hewett, Warburton. 2016. Weight-adjusted DG methods: wave propagation in heterogeneous media (SISC).
Chan, Hewett, Warburton. 2016. Weight-adjusted DG methods: curvilinear meshes (arXiv).
Chan, Warburton 2016. GPU-accelerated Bernstein-Bézier DG methods for wave problems (SISC).

Additional slides

Performance comparisons of BBDG and nodal DG

BBDG: lower FLOPs per second than nodal DG...

but maintains throughput/bandwidth as N increases!

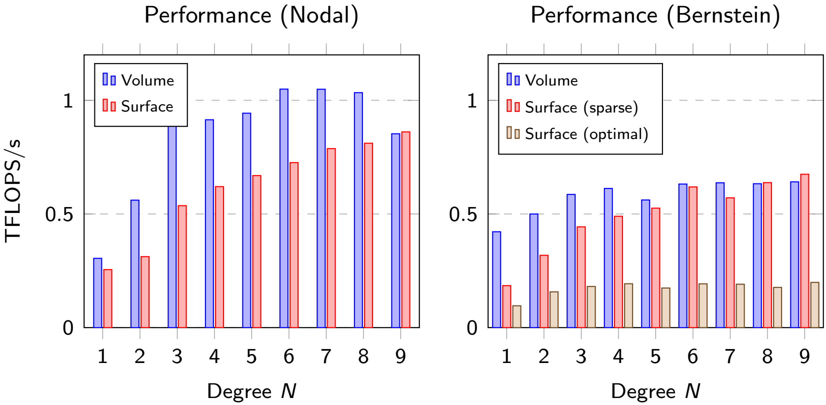


Figure: Profiled FLOPS/s for nodal and Bernstein-Bézier DG methods.

Performance comparisons of BBDG and nodal DG

BBDG: lower FLOPs per second than nodal DG...
but maintains throughput/bandwidth as N increases!

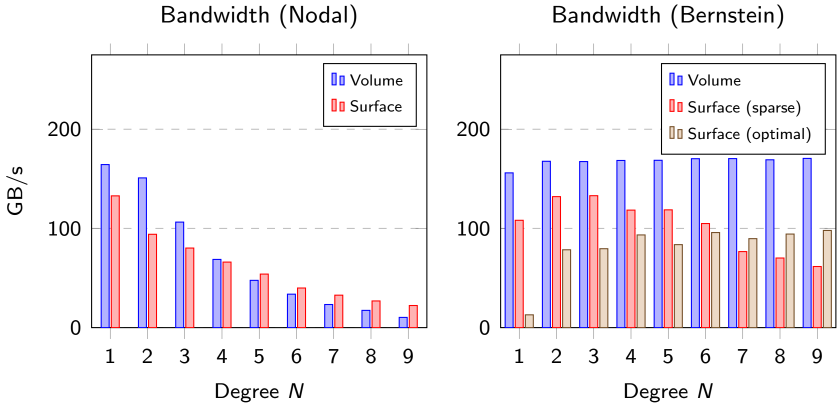
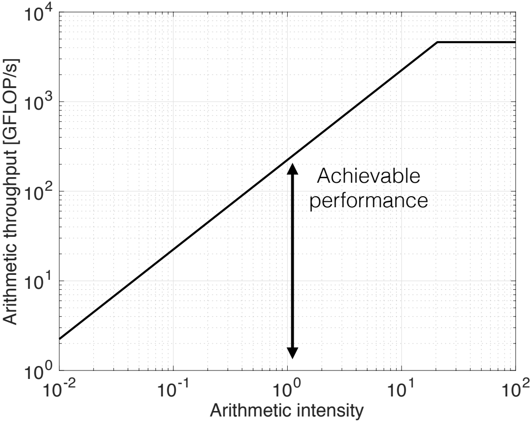


Figure: Profiled bandwidth for nodal and Bernstein-Bézier DG methods.

Roofline model: estimating computational efficiency

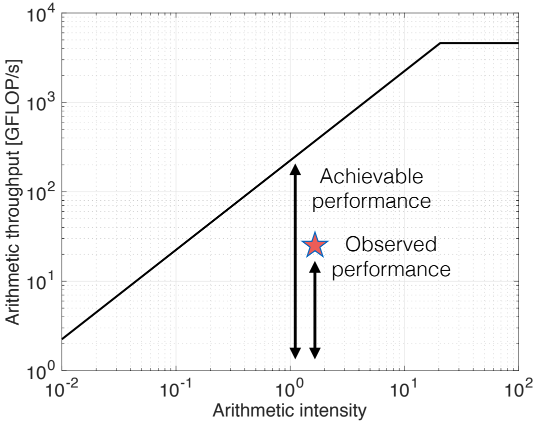
- Arithmetic intensity: floating-point operations per byte of data.
- Computational efficiency: ratio of observed/achievable performance.



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

Roofline model: estimating computational efficiency

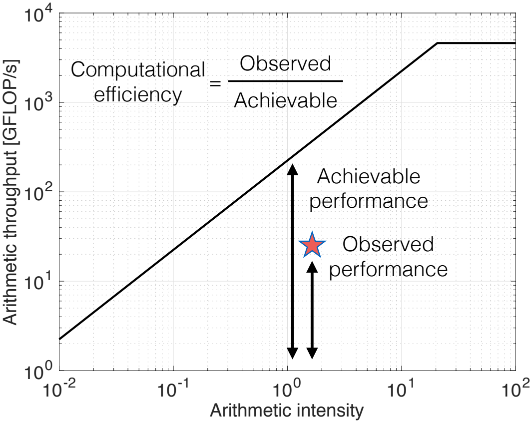
- Arithmetic intensity: floating-point operations per byte of data.
- Computational efficiency: ratio of observed/achievable performance.



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.

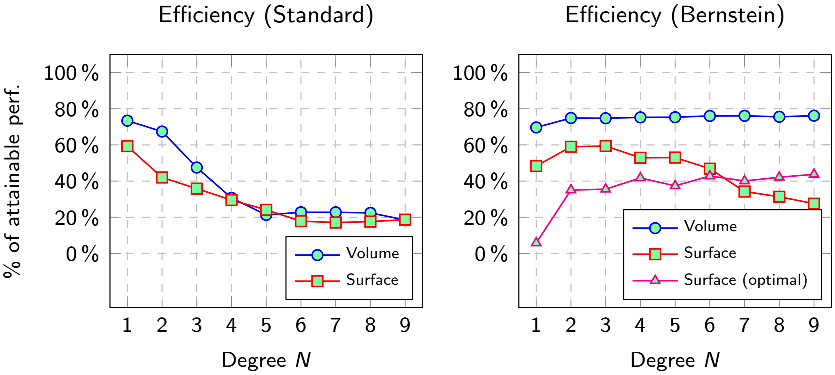


Williams, Waterman, Patterson 2009. Roofline: an insightful visual performance model for multicore architectures.

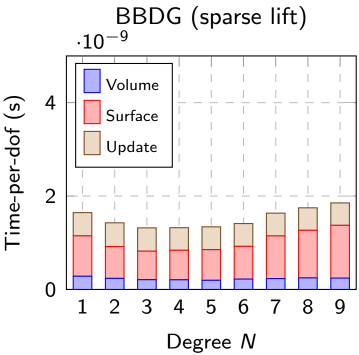
Efficiency comparisons of BBDG and nodal DG

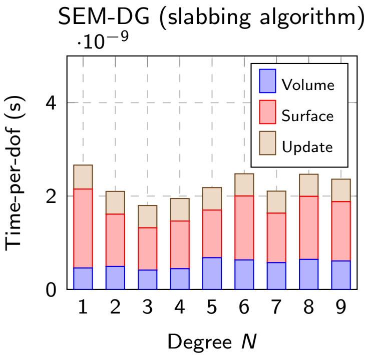
Bernstein-Bézier DG: standard implementation, sparse matrices.

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



Runtime-only comparisons: BBDG, SEM-DG on GPUs

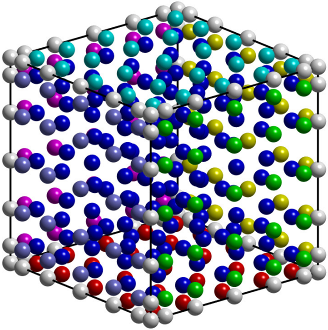




- BBDG 1-1.75$\times$ faster per dof than SEM-DG for $N \leq 10$.
- Unstructured hex meshes: $9(N+1)^3$ geometric factors per element.
- **Disclaimer:** hexes are more accurate, need time-to-error studies!

Abdi, Wilcox, Warburton, Giraldo 2016. A GPU Accel. Cont. and Disc. Galerkin Non-hydrostatic Atmospheric Model

Runtime-only comparisons: BBDG, SEM-DG on GPUs



- BBDG 1-1.75$\times$ 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!

Abdi, Wilcox, Warburton, Giraldo 2016. A GPU Accel. Cont. and Disc. Galerkin Non-hydrostatic Atmospheric Model