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

Jesse Chan, Kaihang Guo

Department of Computational and Applied Math, Rice University

Rice Oil and Gas HPC Conference 2018
March 12-13, 2018
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.

Goal: accuracy and efficiency for heterogeneous media.
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.

Goal: accuracy and efficiency for heterogeneous media.
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.

Coarse quadratic approximation.

Goal: accuracy and efficiency for heterogeneous media.
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.

Goal: accuracy and efficiency for heterogeneous media.
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.

Goal: accuracy and efficiency for heterogeneous media.
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.

Goal: accuracy and efficiency for heterogeneous media.
Time-domain nodal DG methods

Assume \( u(x, t) = \sum u_j \phi_j(x) \) 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).

\[
\frac{du}{dt} = D_x u + \sum_{\text{faces}} L_f (\text{flux}).
\]

\[
M_{ij} = \int_{D^k} \phi_j(x) \phi_i(x)
\]

\[
L_f = M^{-1} M_f.
\]
Time-domain nodal DG methods

Assume $u(x,t) = \sum u_j \phi_j(x)$ 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).

\[
\frac{du}{dt} = D_x u + \sum_{\text{faces}} L_f \text{(flux)}.
\]

Volume kernel

Surface kernel

\[
M_{ij} = \int_{D^k} \phi_j(x) \phi_i(x)
\]

\[
L_f = M^{-1} M_f.
\]
Time-domain nodal DG methods

Assume \( u(x, t) = \sum u_j \phi_j(x) \) 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).

\[
\frac{du}{dt} = D_x u + \sum_{\text{faces}} L_f (\text{flux}).
\]

Volume kernel

Surface kernel

\[
M_{ij} = \int_{D^k} \phi_j(x) \phi_i(x)
\]

\[
L_f = M^{-1} M_f.
\]
Outline

1. Weight-adjusted DG (WADG): arbitrary heterogeneous media

2. Bernstein-Bezier WADG: high order efficiency
Outline

1. Weight-adjusted DG (WADG): arbitrary heterogeneous media

2. Bernstein-Bezier WADG: high order efficiency
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.

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

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

$$\mathbf{M}_{1/c^2} \frac{d\mathbf{p}}{dt} = \mathbf{A}_h \mathbf{U}, \quad (\mathbf{M}_{1/c^2})_{ij} = \int_{D_k} \frac{1}{c^2(x)} \phi_j(x) \phi_i(x).$$
Weight-adjusted DG: stable, accurate, non-invasive

- Weight-adjusted DG (WADG): energy stable approx. of $M_{1/c^2}$

$$M_{1/c^2} \frac{dp}{dt} \approx M (M_{c^2})^{-1} M \frac{dp}{dt} = AhU.$$ 

- New evaluation reuses implementation for constant wavespeed

$$\frac{dp}{dt} = M^{-1} (M_{c^2}) \text{ modified update} \quad \frac{M^{-1} AhU}{M^{-1} A_h U} \text{ constant wavespeed RHS}$$

- Low storage matrix-free application of $M^{-1} M_{c^2}$ using quadrature-based interpolation and $L^2$ projection matrices $V_q, P_q$.

$$(M)^{-1} M_{c^2} \text{RHS} = M^{-1} V_q^T W \text{diag} (c^2) V_q \text{(RHS)}.$$
Weight-adjusted DG (WADG): arbitrary heterogeneous media

WADG: nearly identical to using $M^{-1}_{1/c^2}$

- $L^2$ error is $O(\,h^{N+1})$; standard DG and WADG difference is $O(\,h^{N+2})$.
- Can generalize to matrix weights (elastic wave propagation).

Figure: Standard vs. weight-adjusted DG with spatially varying $c^2$.
Weight-adjusted DG (WADG): arbitrary heterogeneous media

WADG: nearly identical to using $M_{1/c^2}^{-1}$

- $L^2$ error is $O(h^{N+1})$; standard DG and WADG difference is $O(h^{N+2})$.
- Can generalize to matrix weights (elastic wave propagation).

Figure: Standard vs. weight-adjusted DG with spatially varying $c^2$. 
Weight-adjusted DG (WADG): arbitrary heterogeneous media

**WADG: more efficient than storing** $M_{1/c^2}^{-1}$ **on GPUs**

<table>
<thead>
<tr>
<th>$M_{1/c^2}^{-1}$</th>
<th>$N = 1$</th>
<th>$N = 2$</th>
<th>$N = 3$</th>
<th>$N = 4$</th>
<th>$N = 5$</th>
<th>$N = 6$</th>
<th>$N = 7$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>.66</td>
<td>2.79</td>
<td>9.90</td>
<td>29.4</td>
<td>73.9</td>
<td>170.5</td>
<td>329.4</td>
</tr>
<tr>
<td>WADG</td>
<td>0.59</td>
<td>1.44</td>
<td>4.30</td>
<td>13.9</td>
<td>43.0</td>
<td>107.8</td>
<td>227.7</td>
</tr>
<tr>
<td>Speedup</td>
<td>1.11</td>
<td>1.94</td>
<td>2.30</td>
<td>2.16</td>
<td>1.72</td>
<td>1.58</td>
<td>1.45</td>
</tr>
</tbody>
</table>

Time (ns) per element: storing/applying $M_{1/c^2}^{-1}$ vs WADG (deg. $2N$ quadrature).

- **Efficiency on GPUs**: reduce memory accesses and data movement.
- **(Tuned) low storage WADG** faster than storing and applying $M_{1/c^2}^{-1}$!
Computational costs at high orders of approximation

Problem: WADG at high orders becomes expensive!

WADG runtimes for 50 timesteps, 98304 elements.

- Large dense matrices: $O(N^6)$ work per tet.
- High orders usually use tensor-product elements: $O(N^4)$ vs $O(N^6)$ cost, but less geometric flexibility.
- Idea: choose basis such that matrices are sparse.
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.
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.

Bernstein-Bezier WADG: high order efficiency

**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.

---

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.

---

Bernstein-Bezier WADG: high order efficiency

BBDG: efficient volume, surface kernels

Nodal DG

\[ \frac{\text{d} \mathbf{u}}{\text{d} t} = D_x \mathbf{u} + \sum_{\text{faces}} L_f (\text{flux}), \]

\[ L_f = M^{-1} M_f. \]

Bernstein-Bezier DG

Degree $N$
**BBDG: efficient volume, surface kernels**

\[
\frac{du}{dt} = D_x u + \sum_{\text{faces}} L_f (\text{flux}), \quad L_f = M^{-1} M_f.
\]

**Update kernel**

**Volume kernel**

**Surface kernel**

**BBDG speedup over nodal DG**
BBWADG: polynomial multiplication and projection

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

- WADG: can reuse fast Bernstein volume and surface kernels.
- $O(N^6)$ update kernel: $V_q$ interpolates $u(x)$ to quadrature points, scale by $c^2(x)$ at quadrature points, apply $P_q$ to project back to $P^N$.
- New approach: approx. $c^2(x)$ with degree $M$ polynomial, use fast Bernstein algorithms for polynomial multiplication and projection.
Fast Bernstein polynomial multiplication

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

- Given $c^2(x)u(x)$ as a degree $(N + M)$ polynomial, apply $L^2$ projection matrix $P_{N}^{N+M}$ to reduce to degree $N$.

- Polynomial $L^2$ projection matrix $P_{N}^{N+M}$ under Bernstein basis:

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

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

$$
\begin{pmatrix}
    c_0 I + E_{N-1}^N & \left( c_1 I + E_{N-2}^{N-1} (c_2 I + \cdots) \right) \left( E_{N-2}^{N-1} \right)^T
\end{pmatrix}
\begin{pmatrix}
    E_{N-1}^N
\end{pmatrix}^T
$$
Sketch of GPU algorithm for $\tilde{P}_N$

$O(N^3)$ threads

$O(N^3)$ shared memory

$O(N)$ register memory per thread

$$
\begin{pmatrix}
c_0 I + E_{N-1}^N & \left(c_1 I + E_{N-2}^{N-1} \left(c_2 I + \cdots \right) \left(E_{N-2}^{N-1}\right)^T \right)
\end{pmatrix}
\left(E_{N-1}^N\right)^T
$$
Approximating smooth $c^2(x)$ using $L^2$ projection:

- $O(h^2)$ for $M = 0$,
- $O(h^4)$ for $M = 1$,
- $O(h^{M+3})$ for $0 < M \leq N - 2$. 

### Diagram Details

- **Mesh size $h$**
- **L2 error**
- **BBWADG**
- **WADG ($N = 4$)**

- **Piecewise constant $c^2$**
- **Piecewise linear $c^2$**
- **Quadratic $c^2$**
BBWADG: computational runtime (acoustics)

Update kernel for $M = 1$: runtime per element

![Graph showing runtime comparison between BBWADG and WADG for different degrees $N$. The graph plots runtime against degree $N$ on a log-log scale, showing exponential growth for both methods. BBWADG is indicated by red triangles, WADG by blue triangles, $N^4$ by red squares, and $N^6$ by blue squares.](image-url)
BBWADG: update kernel speedup over WADG (acoustics)

<table>
<thead>
<tr>
<th></th>
<th>$N = 3$</th>
<th>$N = 4$</th>
<th>$N = 5$</th>
<th>$N = 6$</th>
<th>$N = 7$</th>
<th>$N = 8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>WADG</td>
<td>1.60e-8</td>
<td>3.34e-8</td>
<td>6.94e-8</td>
<td>1.28e-7</td>
<td>3.31e-7</td>
<td>3.03e-6</td>
</tr>
<tr>
<td>BBWADG</td>
<td>2.20e-8</td>
<td>3.30e-8</td>
<td>4.42e-8</td>
<td>6.01e-8</td>
<td>9.46e-8</td>
<td>1.31e-7</td>
</tr>
<tr>
<td>Speedup</td>
<td>0.7260</td>
<td>1.0127</td>
<td>1.5706</td>
<td>2.1258</td>
<td>3.4938</td>
<td>23.1591</td>
</tr>
</tbody>
</table>

For $N \geq 8$, quadrature (and WADG) becomes much more expensive.
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?