Preconditioning of Newton-GMRES solvers for Discontinuous Galerkin Problems

Per-Olof Persson

Massachusetts Institute of Technology


BIRS Workshop on Discontinuous Galerkin Methods for Partial Differential Equations

Banff, November 29, 2007
Motivation

- Need for higher fidelity CFD prediction
  - DNS/LES/DES applications
  - Accurate RANS for engineering applications (drag prediction, rotor dynamics, fluid/structure interaction, flapping flight)
  - Computational aeroacoustics (direct solution of compressible flow, accurate computation of noise sources)
Motivation

• Three fundamental properties:

<table>
<thead>
<tr>
<th></th>
<th>FVM</th>
<th>FDM</th>
<th>FEM</th>
<th>DG</th>
</tr>
</thead>
<tbody>
<tr>
<td>High-order/Low dispersion</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Unstructured meshes</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Stability for conservation laws</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
</tr>
</tbody>
</table>

• However, several problems to resolve:
  – High CPU/memory requirements (compared to FVM or H-O FDM)
  – Low tolerance to under-resolved features
  – High-order geometry representation and mesh generation

• The challenge is to make DG methods competitive for real-world problems
The Compact Discontinuous Galerkin Method

- Modification of LDG fluxes for compact stencil
- Strictly fewer connectivities than LDG and BR2
- More accurate than LDG and BR2
The Local DG Method

- Consider Poisson problem \(-\nabla \cdot (\kappa \nabla u) = f\)
- Write as system of first order equations,
  \[-\nabla \cdot \sigma = f\]
  \[\sigma = \kappa \nabla u\]
- Use numerical inter-element fluxes
  \[\hat{\sigma} = \{\sigma_h\} - C_{11}[u_h] + C_{12}[\sigma_h]\]
  \[\hat{u} = \{u_h\} - C_{12} \cdot [u_h]\]
- Choose a \textit{switch} which satisfies \(S_{K+}^{K} + S_{K-}^{K} = 1\) and set
  \[C_{12} = \frac{1}{2} (S_{K+}^{K} n^+ + S_{K-}^{K} n^-)\]
- In general, this will introduce non-local couplings
The Local DG Scheme

- For the LDG scheme, the variables $\sigma_h$ can be written as

$$
\sigma_h = \kappa \nabla_h u_h + \bar{\sigma}_h
$$

where

$$
\bar{\sigma}_h = \kappa r([u_h]) + \kappa l(C_{12} \cdot [u_h]) + \text{boundary terms}
$$

and $r(\phi)$ and $l(q)$ are lifting operators.

- In general, this may introduce non-local couplings since the lifting operators involve all element edges.
The Compact DG Scheme

- In our CDG scheme, we split the lifting operators into sums of edge-wise lifting operators $r^e(\phi), l^e(q)$, and set

$$\hat{\sigma} = \{\sigma^e_h\} - C_{11}[u_h] + C_{12}[\sigma^e_h]$$

$$\hat{u} = \{u_h\} - C_{12} \cdot [u_h]$$

where $\sigma^e_h = \kappa \nabla_h u_h + \bar{\sigma}^e_h$, with

$$\bar{\sigma}^e_h = \kappa r^e([u_h]) + \kappa l^e(C_{12} \cdot [u_h]) + \text{boundary terms}$$

- Since only the lifting operator corresponding to the current edge is used, only neighboring elements are connected
Error Estimates

• Written in primal form, the LDG scheme becomes (ignoring bnd terms):

\[ \int_{\Omega} \kappa \left( r([u]) + l(C_{12} \cdot [u]) \right) \cdot \left( r([v]) + l(C_{12} \cdot [v]) \right) dx = \]

\[ \sum_{e \in E_i} \sum_{f \in E_i} \int_{\Omega} \kappa \left( r^e([u]) + l^e(C_{12} \cdot [u]) \right) \cdot \left( r^f([v]) + l^f(C_{12} \cdot [v]) \right) dx \]

• The CDG scheme is identical except for some excluded indefinite terms:

\[ \sum_{e \in E_i} \int_{\Omega} \kappa \left( r^e([u]) + l^e(C_{12} \cdot [u]) \right) \cdot \left( r^e([v]) + l^e(C_{12} \cdot [v]) \right) dx = \]

\[ \sum_{e \in E_i} \sum_{f \in E_i} \delta_{ef} \int_{\Omega} \kappa \left( r^e([u]) + l^e(C_{12} \cdot [u]) \right) \cdot \left( r^f([v]) + l^f(C_{12} \cdot [v]) \right) dx \]

• Non-compact connections are eliminated but the scheme remains stable
Error Estimates

• Coercivity and boundedness same as LDG, leading to a-\textit{priori} estimates:

\[ |||u - u_h||| \leq Ch^p |u|_{p+1, \Omega} \]

and

\[ ||u - u_h||_{0, \Omega} \leq Ch^{p+1} |u|_{p+1, \Omega}. \]

with the norm

\[ |||v|||^2 = \sum_{K \in T_h} |v|_{1,K}^2 + \sum_{e \in E_i} ||r_e([v])||_{0, \Omega}^2 + \sum_{e \in \partial \Omega_D} ||r_D(v)||_{0, \Omega}^2 \]

• Assumes \( C_{11} = \mathcal{O}(h^{-1}) \), but is observed numerically for \( C_{11} = 0 \)
Matrix Representation

- Block matrix representation *fundamental for high performance*
  - Solver algorithms based on blocks
  - Up to 10 times higher performance with optimized BLAS

- Compact stencil $\implies$ Matrix structure given by mesh connectivities

- Harder to store LDG/BR2/IP efficiently (many use full block storage)

CDG – 2 arrays

LDG – 3 arrays + struct

BR2 / IP – 3 arrays
Switches and Null-space Dimensions

- Unlike the LDG scheme, the CDG scheme appears to be stable for $C_{11} = 0$ and an *inconsistent switch* such as highest element number.

- Simple test [Sherwin et al 05]: Poisson problem, periodic boundary conditions, expected nullspace dimension = 1

<table>
<thead>
<tr>
<th>Nullspace dimension</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polynomial order $p$</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>Consistent switch</td>
<td>CDG</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>LDG</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Natural switch</td>
<td>CDG</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>LDG</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>
ILU and Switch Orientation

- Orientation of lower-triangular blocks important for ILU sparsity
- Take advantage of CDG’s insensitivity to orientation

Switch 1: 
Same LU storage

Switch 2: 
More LU storage
Switches and Null-space Dimensions

- No additional non-zeros in block-ILU(0) factorization using CDG
- Dense lower-triangular blocks using BR2 / IP

**CDG**

<table>
<thead>
<tr>
<th>Stiffness Matrix</th>
<th>Block ILU(0)</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Stiffness Matrix" /></td>
<td><img src="image" alt="Block ILU(0)" /></td>
</tr>
</tbody>
</table>

640 non-zeros

640 non-zeros
Switches and Null-space Dimensions

- No additional non-zeros in block-ILU(0) factorization using CDG
- Dense lower-triangular blocks using BR2 / IP

### BR2 / IP

<table>
<thead>
<tr>
<th>Stiffness Matrix</th>
<th>Block ILU(0)</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Matrix Visualization" /></td>
<td><img src="image2.png" alt="Matrix Visualization" /></td>
</tr>
</tbody>
</table>

- Stiffness Matrix: 784 non-zeros
- Block ILU(0): 892 non-zeros
Preconditioners for Krylov Methods

- Performance of Krylov methods greatly improved by preconditioning

- Various standard methods:
  - Block-diagonal: \( \mathbf{A} \approx \text{blockdiag}(\mathbf{A}) \)
    - Poor in general
  - Block-Incomplete LU: \( \mathbf{A} \approx \tilde{\mathbf{L}}\tilde{\mathbf{U}} \)
    - Good for convection (with the right ordering, more later)
  - \( p \)-multigrid (low-degree) preconditioner
    - Good for diffusion

- Highly efficient combination [Persson/Peraire 06]
  - Use block-ILU(0) as post-smoother for coarse scale correction
  - Combines advantages of ILU and low-degree preconditioners
  - Cheap general purpose preconditioner
Simple Preconditioners

- Block-diagonal (Jacobi):

\[
\tilde{A}^J_{ij} = \begin{cases} 
A_{ij} & \text{if } i = j, \\
0 & \text{if } i \neq j.
\end{cases}
\]

- Block-Gauss Seidel

\[
\tilde{A}^{GS}_{ij} = \begin{cases} 
A_{ij} & \text{if } i \leq j, \\
0 & \text{if } i > j.
\end{cases}
\]

- Good performance for certain problems, but generally too slow
Incomplete LU Factorization

- For our low-degree dual meshes, it is uncommon that an element \( k \) is a neighbor of both element \( j \) and \( i \) when \( j \) is a neighbor of \( i \)

- This simplifies the Block-ILU(0) algorithm:

\[
\tilde{U} \leftarrow A, \quad \tilde{L} \leftarrow I
\]

\[
\text{for } j = 1 \text{ to } n - 1
\]

\[
\text{for neighbors } i > j \text{ of } j
\]

\[
\tilde{L}_{ij} = \tilde{U}_{ij} \tilde{U}_{jj}^{-1}
\]

\[
\tilde{U}_{ii} \leftarrow \tilde{U}_{ii} - \tilde{L}_{ik} \tilde{U}_{ki}
\]

end for

end for

- This means \( \tilde{U}_{ij} = A_{ij} \) when \( j > i \), or \( \tilde{U} \) only differs from \( A \) in the diagonal blocks

- Store only diagonal blocks of \( \tilde{U} \)
Computational Cost

- Pre-calculation cost (block-size $N$, dimension $D$, $ne$ simplex elements):

<table>
<thead>
<tr>
<th>Operation</th>
<th>Flop count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jacobi Factorization $\tilde{A}^J$</td>
<td>$(2/3)N^3ne$</td>
</tr>
<tr>
<td>Gauss Seidel Factorization $\tilde{A}^{GS}$</td>
<td>$(2/3)N^3ne$</td>
</tr>
<tr>
<td>ILU(0) Factorization $\tilde{A}^{ILU}$</td>
<td>$(2D + 8/3)N^3ne$</td>
</tr>
</tbody>
</table>

- Notes: This assumes full blocks (CDG is significantly cheaper) and cost of assembling a nonlinear $A$ is also $O(N^3ne)$ with a larger constant

- Cost per iteration:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Flop count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jacobi solve $(\tilde{A}^J)^{-1}p$</td>
<td>$(2)N^2ne$</td>
</tr>
<tr>
<td>Gauss Seidel solve $(\tilde{A}^{GS})^{-1}p$</td>
<td>$(D + 3)N^2ne$</td>
</tr>
<tr>
<td>ILU(0) solve $(\tilde{A}^{ILU})^{-1}p$</td>
<td>$(2D + 4)N^2ne$</td>
</tr>
<tr>
<td>Matrix-vector product $Ax$</td>
<td>$(2D + 4)N^2ne$</td>
</tr>
</tbody>
</table>

- Note: Only factor of 2 between ILU + MATVEC and Jacobi + MATVEC
Minimum Discarded Fill Ordering

- Performance of ILU(0) highly dependent of ordering

- Greedy algorithm for element ordering [Persson/Peraire 07]:
  At step $j$, if $j'$ is chosen next, we would discard the fill
  \[
  \Delta \tilde{U}_{ik}^{(j,j')} = -\tilde{U}_{ij'}\tilde{U}_{j'j}\tilde{U}_{j'k}, \quad \text{for neighbors } i \geq j, k \geq j \text{ of element } j'
  \]

- Choose the $j'$ that minimizes the norm of the discarded fill
  \[
  w^{(j,j')} = \| \Delta \tilde{U}^{(j,j')} \|_F \leq \| \tilde{U}_{ij'} \|_F \| \tilde{U}_{j'j}\tilde{U}_{j'k}^{-1} \|_F
  \]

- Some simplifications and a min-heap data structure $\implies \mathcal{O}(n \log n)$ computational cost

- Similar to the Minimum Degree algorithm, but considering the magnitude of the fill instead of just the size
Minimum Discard Fill Ordering

\[ B \leftarrow (\tilde{A}^J)^{-1} A \]
\[ C_{ij} \leftarrow \|B_{ij}\|_F \]

for \( k = 1, \ldots, n \)
\[ \Delta C \leftarrow 0 \]

for neighbors \( i, j \) of element \( k, i \neq j \)
\[ \Delta C_{ij} \leftarrow C_{ik}C_{kj} \]

end for
\[ w_k \leftarrow \|\Delta C\|_F \]

end for

for \( i = 1, \ldots, n \)
\[ p_i \leftarrow \text{argmin}_j w_j \]
\[ w_{p_i} \leftarrow \infty \]

for neighbors \( k \) of \( p_i \) not yet numbered
    Recompute \( w_k \), only considering neighbors not yet numbered
end for

end for
Effect of Ordering on Convection-Diffusion

- Convection-Diffusion model problem, with $(\alpha, \beta) = (1, 2x), \varepsilon \geq 0$:

\[
\frac{\partial u}{\partial t} + \nabla \cdot \begin{bmatrix} \alpha u \\ \beta u \end{bmatrix} - \nabla \cdot \begin{bmatrix} \varepsilon u_x \\ \varepsilon u_y \end{bmatrix} = 0
\]
Effect of Ordering on Convection-Diffusion

- Reverse Cuthill-McKee vs. Minimum Discarded Fill element ordering
- MDF perfect for convection, but small variations for diffusion
The ILU(0)-p1 Preconditioner

- Combination of block ILU and multigrid [Persson/Peraire 06]

- Coarse scale correction + post-smoothing by $\tilde{A} = \tilde{L}\tilde{U}$:
  
  0. $A^{(0)} = P^T AP$  \textit{Precompute coarse operator, block wise}
  1. $b^{(0)} = P^T b$  \textit{Restrict residual element/component wise}
  2. $A^{(0)} u^{(0)} = b^{(0)}$  \textit{Solve coarse scale problem}
  3. $u = Pu^{(0)}$  \textit{Prolongate solution element/component wise}
  4. $u = u + \alpha \tilde{A}^{-1}(b - Au)$  \textit{Apply smoother $\tilde{A}$ with damping $\alpha$}

- Restriction/prolongation operator $P$ block diagonal, based on orthogonal Koornwinder polynomials

- Coarse scale problem solved directly (2-D problems, serial) or iteratively by GMRES or $h$-multigrid
The ILU-p1 Preconditioner - Convection-Diffusion

- Block ILU perfect for convection, multigrid perfect for diffusion
- Block ILU-smoothed multigrid (BILU0-p1) almost perfect for any $\varepsilon$
Effect of Ordering on Navier-Stokes

- Model Navier-Stokes problem for large range of Reynolds numbers
## The ILU-p1 Preconditioner – Navier-Stokes

<table>
<thead>
<tr>
<th>Problem</th>
<th>Parameters</th>
<th>Preconditioner/Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\Delta t$</td>
<td>$M$</td>
</tr>
<tr>
<td></td>
<td>$\times$</td>
<td>$\times$</td>
</tr>
<tr>
<td>Inviscid</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$10^{-3}$</td>
<td>0.2</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>2</td>
</tr>
<tr>
<td>$\infty$</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>$10^{-1}$</td>
<td>0.2</td>
<td>187</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>$\infty$</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>Laminar</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$10^{-3}$</td>
<td>0.2</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>5</td>
</tr>
<tr>
<td>$\infty$</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>Re=1,000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$10^{-3}$</td>
<td>0.2</td>
<td>597</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>$\infty$</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>$10^{-1}$</td>
<td>0.2</td>
<td>98</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>$\infty$</td>
<td>0.1</td>
<td>1</td>
</tr>
</tbody>
</table>
## The ILU-p1 Preconditioner – Navier-Stokes

<table>
<thead>
<tr>
<th>Problem</th>
<th>Parameters</th>
<th>Preconditioner/Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\Delta t$</td>
<td>$M$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Laminar</td>
<td>$10^{-3}$</td>
<td>0.2</td>
</tr>
<tr>
<td>Re=20,000</td>
<td>$10^{-1}$</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>$\infty$</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>$10^{-3}$</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>$10^{-1}$</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>$\infty$</td>
<td>0.01</td>
</tr>
<tr>
<td>RANS</td>
<td>$10^{-3}$</td>
<td>0.2</td>
</tr>
<tr>
<td>Re=10^6</td>
<td>$10^{-1}$</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>$\infty$</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>$10^{-3}$</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>$10^{-1}$</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>$\infty$</td>
<td>0.01</td>
</tr>
</tbody>
</table>
Effect of Ordering on Navier-Stokes

- Good element ordering critical for Block-ILU/multigrid solver

- For $M = 0.2$, $\Delta t = 1.0$:

<table>
<thead>
<tr>
<th>Problem</th>
<th>Element Ordering</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Random</td>
</tr>
<tr>
<td>Inviscid</td>
<td>51</td>
</tr>
<tr>
<td>Laminar, Re=1,000</td>
<td>200</td>
</tr>
<tr>
<td>Laminar, Re=20,000</td>
<td>197</td>
</tr>
<tr>
<td>RANS, Re=$10^6$</td>
<td>98</td>
</tr>
</tbody>
</table>
Mach Number Dependence

- With BILU0-p1, convergence is almost independent of Mach number.
Conclusions

- Important contributions for improving the performance of implicit DG:
  - Efficient viscous discretization (the CDG method)
  - General purpose multigrid/block-ILU preconditioner
  - Matrix-based ILU element ordering by minimum discarded fill

- Current work: New sparser discretizations, 3-D curved mesh generation, orderings for parallel ILU, extensions for LES/DES, structural dynamics, applications in flapping flight, aeroacoustics, and transonic flows